

Sismio

Membri del team di sviluppo:

Matteo Pellegrino 0000766387

Alberto Bagnacani 0000767392

Federico Terzi 0000753482

Sommario:

Abstract	5
Analisi dei requisiti	6
Requisiti del sistema	6
Analisi del dominio	7
Vocabolario	7
Sistemi esterni	8
Analisi dei requisiti	9
Casi d'uso	9
Modello	9
Scenari	11
Analisi del rischio	20
Valutazione dei beni	20
Analisi minacce e controlli	20
Analisi della tecnologia dal punto di vista della sicurezza	21
Security Use Case e Misuse Case	23
Requisiti di Protezione dei Dati	29
Descrizioni delle interfacce grafiche	30
Struttura	30
Schermata di accesso	30
Dashboard	31
Gestione stazioni remote	32
Storico eventi	32
Gestione account	33
Analisi del problema	34
Analisi del documento dei requisiti	34
Analisi delle funzionalità	34
Analisi dei vincoli	37
Tabella dei vincoli	37
Analisi delle interazioni	39
Tabella maschere	39
Tabella sistemi esterni	40
Analisi dei ruoli e delle responsabilità	40
Utente: Tabella Ruolo-Informazioni	41
Amministratore: Tabella Ruolo-Informazioni	41
Scomposizione del problema	41
Creazione modello del dominio	43
Architettura logica	44

Struttura	44
Diagramma dei package	44
Diagramma delle classi	45
Interazione	49
Comportamento	52
Piano del lavoro	53
Sviluppi futuri	53
Piano di collaudo	54
Progetto	57
Progettazione architetturale	57
Requisiti non funzionali	57
Scelta dell'architettura	57
Cliente	57
Servitore	58
Persistenza	58
Considerazioni sulla sicurezza relative alle tecnologie utilizzate	59
Database	59
Trasmissione remota dei dati	59
Progettazione di Dettaglio	61
Struttura	61
Package io.sismio.sensore	61
Package io.sismio.trasmissione	65
Package io.sismio.analisi	67
Package io.sismio.database	69
Package io.sismio.evento	70
Package io.sismio.stazione	73
Package io.sismio.utente	74
Breve considerazione sulla sicurezza	74
Package io.sismio.log	76
Diagramma di Dettaglio - GestioneUtenti	77
Diagramma di Dettaglio - GestioneStazioni	78
Diagramma di Dettaglio - Storico	79
Diagramma di Dettaglio - Dashboard	81
Interazione	82
Creazione della connessione sicura	82
Comportamento	84
Algoritmo Analisi Magnitudo	84
Algoritmo Analisi Frequenza	85
Persistenza	86
Sicurezza	86
Formato del file di log	87

Protezione del file di log	87
Progettazione del collaudo	88
Piano del deployment	91
Implementazione	93
Deployment	94
Deployment Specification	94
Artefatti	94
Deployment Type-Level	95

Abstract

Il progetto intende creare uno strumento dinamico e scalabile per la rilevazione, analisi e gestione di scosse sismiche nel territorio.

La distribuzione dell'applicativo permette la condivisione di dati rilevati mediante sorgenti esterne, quali possono essere sensori hardware, direttamente collegati al terminale oppure un flusso di dati remoto proveniente da un altro nodo del sistema.

L'analisi dei dati sfrutterà un'interfaccia comune ai vari moduli per offrire la possibilità di aggiungere e modificare l'implementazione dei vari componenti e le funzionalità del sistema.

La necessità dei brevi tempi di risposta implica l'utilizzo della concorrenza.

La riservatezza dei dati verrà permessa grazie ad un sistema di autenticazione dotato di un meccanismo di crittografia.

Si vuole offrire una struttura di allarmistica di tipo testuale, visuale e sonora, notificando l'utente in maniera tempestiva in seguito a situazioni critiche rilevate.

Sarà possibile consultare lo storico degli eventi rilevati con funzionalità di ricerca e filtraggio.

L'interfaccia grafica garantisce una visualizzazione semplice, intuitiva e variegata valorizzando la comprensione di dati sensibili mediante l'uso di diagrammi e tabelle.

Analisi dei requisiti

Requisiti del sistema

- Il sistema real-time deve rispondere ad un *evento sismico* registrato entro un limite di tempo prefissato riportando data, ora, frequenza o magnitudo, priorità e luogo dell'avvenimento. La scelta di questi parametri è scaturita in seguito ad un incontro con un esperto del settore, il quale ha inoltre definito diverse tipologie di analisi da effettuare sui dati rilevati: analisi sulla frequenza e sulla magnitudo. E' possibile inoltre visualizzare direttamente i valori letti dal sensore
- Le priorità specificate sono d'informazione (info, magnitudo 0-2 scala Richter o frequenza 1 scossa/ora), di avviso (warning, magnitudo 3-5 scala Richter o frequenza 2 scosse/ora), di allerta (alert, magnitudo 6-7 scala Richter o frequenza 3 scosse/ora), critiche (critical, magnitudo 8-9 scala Richter, frequenza 4 scosse/ora) e fatali (fatal, magnitudo 10 scala Richter, frequenza 5+ scosse/ora)
- Al termine di un evento vengono registrate tutte le informazioni a riguardo. Lo *storico* permette la lettura di tutti gli eventi registrati, mostrandoli a video tramite un'opportuna interfaccia grafica
- L'utente deve poter consultare lo *storico* degli eventi permettendo azioni di ricerca testuale e di filtraggio in base a priorità, data e ora
- L'utente deve poter controllare continuamente le analisi in atto tramite un'unica interfaccia Dashboard contenente tutti i grafici relativi
- Il sistema allarmistico risponde a determinati eventi prioritari informando l'utente tramite *notifiche* variegata, che possono essere sonore, visuali e testuali
- Il sistema trae informazioni da *sensori* locali o remoti in modo *trasparente*
- La *modularità* del sottosistema di analisi permette di aggiungere e modificare con facilità le elaborazioni effettuate sui dati, in base a necessità future del cliente
- Il sistema è un modello distribuito *autocontenuto*, dove ogni *nodo* può essere cliente e/o servitore di un altro nodo, permettendo il collegamento ad una *stazione* remota
- Ogni stazione può avere al massimo un sensore
- La crittografia permetterà la protezione dei dati in sinergia con un sistema di autenticazione basato su utenti con diversi *privilegi*
- I privilegi sono basati su due tipi di account: Utente e Amministratore. Quest'ultimo possiede tutti i privilegi dell'Utente oltre alle autorizzazioni per la registrazione ed eliminazione degli utenti e delle stazioni
- L'Amministratore deve poter consultare la lista delle stazioni e degli utenti con la possibilità di effettuare azioni di ricerca
- Per accedere ad una stazione un *utente* dovrà fornire una coppia di *credenziali*, formata da *username* e *password*

Analisi del dominio

Vocabolario

Voce	Definizione	Sinonimi
Sensore	Dispositivo meccanico, elettronico o chimico in grado di rilevare una grandezza fisica e di trasmettere le variazioni a un sistema di misurazione o di controllo	Strumento di misura
Real-time	Il sistema garantisce tempi di risposta rapidi	
Storico	Registro di rilevazioni sismiche presenti e passate	
Evento sismico	Rilevazione di notevole interesse	Avvenimento, terremoto, scossa sismica, dato
Notifica	Avviso da parte del sistema nei confronti degli utenti alla rilevazione di <i>eventi sismici</i>	Comunicazione
Stazione	Terminale fisico a cui è collegato un <i>sensore</i>	Nodo
Autocontenuto	Ogni <i>stazione</i> può inviare e/o ricevere informazioni	
Trasparente	La modalità di scambio delle informazioni risulta impercettibile all'utente	
Modularità	Concetto architetturale che permette la modifica, creazione e cancellazione di funzionalità in modo semplice e robusto	
Privilegi	Insieme di azioni che un utente può o non può effettuare	Autorizzazioni
Utente	Utilizzatore di una <i>stazione</i>	Persona

Credenziali	Insieme composto da username e password, necessari per accedere ad una stazione	
Username	Stringa alfanumerica	
Password	Codice alfanumerico	

Sistemi esterni

Ogni stazione viene interfacciata con un sensore esterno in grado di rilevare le vibrazioni del terreno in tempo reale. Il sensore, realizzato utilizzando Arduino, comunica con la stazione attraverso una porta seriale.

La relazione tra il caso d'uso RegistrazioneUtente e l'attore Utente nasce dal fatto che questo affianca l'Amministratore durante l'inserimento dati.

Scenari

Titolo	GestioneEventi
Descrizione	Lettura dei dati dal Sensore, con conseguente analisi e notificazione all'utente.
Attori	Sensore, EventoSismico
Relazioni	
Precondizioni	Si verifica un evento sismico.
Postcondizioni	Il sistema ha rilevato l'EventoSismico L'Utente è stato notificato dell'EventoSismico rilevato Il sistema ha registrato l'EventoSismico in maniera persistente.
Scenario principale	<ol style="list-style-type: none"> 1. Il Sensore invia i dati a GestioneEventi 2. GestioneEventi effettua le analisi relative a Magnitudo e Frequenza delle scosse sismiche. 3. A seguito delle analisi, viene rilevato un EventoSismico. 4. Il sistema notifica l'Utente dell'EventoSismico in modo visivo, testuale e sonoro. 5. GestioneEventi si occupa di registrare l'EventoSismico in maniera persistente. 6. Il sistema continua la sua elaborazione, analizzando il prossimo istante di tempo e ripetendo la procedura.
Scenari alternativi	<p>Scenario a: La connessione con il Sensore viene interrotta.</p> <ol style="list-style-type: none"> 1. Viene mostrato a video un messaggio d'errore. <p>Scenario b: Si verifica un errore nell'analisi dei dati</p> <ol style="list-style-type: none"> 1. Il sistema notifica l'utente dell'errore 2. Il sistema cerca di recuperare la situazione nell'istante successivo di tempo.
Requisiti non funzionali	<p>Integrità dei dati letti dal Sensore</p> <p>Integrità degli EventiSismici salvati in maniera persistente</p> <p>Analisi veloce ed efficiente</p> <p>Rapidità lettura e scrittura dati</p> <p>Notificazione dell'EventoSismico all'utente in maniera efficace e tempestiva</p>
Punti aperti	

Titolo	Storico
Descrizione	Il sistema offre all'utente l'elenco degli EventiSismici registrati
Attori	Utente
Relazioni	Autenticazione, FiltroStorico
Precondizioni	
Postcondizioni	Il sistema ha mostrato all'Utente gli EventiSismici registrati in maniera persistente.
Scenario principale	<ol style="list-style-type: none"> 1. Autenticazione 2. Utente seleziona la schermata relativa allo Storico 3. Viene mostrata una schermata contenente tutti gli eventi 4. L'utente può scegliere se filtrare gli EventiSismici tramite FiltroStorico premendo il pulsante relativo
Scenari alternativi	<p>Scenario a: L'Utente non è autenticato o la sessione è scaduta</p> <ol style="list-style-type: none"> 1. Autenticazione <p>Scenario b: Lo Storico non contiene eventi</p> <ol style="list-style-type: none"> 1. Viene mostrato a video un avviso all'utente
Requisiti non funzionali	Facilità di consultazione degli EventiSismici registrati in maniera persistente.
Punti aperti	

Titolo	FiltroStorico
Descrizione	Il sistema permette all'utente di filtrare gli EventiSismici visualizzati nello Storico tramite diversi parametri di ricerca.
Attori	Utente
Relazioni	Storico
Precondizioni	
Postcondizioni	Il sistema ha mostrato all'Utente gli EventiSismici persistenti che rispettano i parametri specificati.
Scenario principale	<ol style="list-style-type: none"> 1. Il sistema mostra una schermata all'Utente relativa ai vari parametri di ricerca impostabili. 2. L'Utente seleziona uno o più parametri di ricerca, a scelta tra: priorità, data e ora o ricerca di testo. 3. Il sistema effettua la ricerca di tutti gli EventiSismici che rispettano

	i parametri di ricerca specificati. 4. Il sistema mostra all'Utente gli EventiSismici trovati.
Scenari alternativi	Scenario a: Non sono stati trovati EventiSismici che rispettano i parametri di ricerca specificati. 1. Viene mostrato un avviso all'utente
Requisiti non funzionali	Facilità di inserimento dei parametri di ricerca e velocità di elaborazione del sistema. Rapidità di ricerca.
Punti aperti	

Titolo	Autenticazione
Descrizione	Modalità di accesso al sistema da parte dell'Utente
Attori	Utente
Relazioni	Storico, GestioneUtenti, GestioneStazioni
Precondizioni	
Postcondizioni	L'Utente è autenticato presso il sistema
Scenario principale	<ol style="list-style-type: none"> 1. Il sistema presenta la schermata di accesso all'Utente 2. L'Utente inserisce le proprie credenziali 3. L'Utente preme il bottone di accesso 4. Il sistema verifica le credenziali e queste risultano corrette 5. Viene mostrata la schermata principale
Scenari alternativi	<p>Scenario a: Le credenziali non sono valide</p> <ol style="list-style-type: none"> 1. Viene mostrato un avviso all'utente 2. Il sistema mostra nuovamente la schermata di accesso al sistema <p>Scenario b: L'username non è presente nel sistema</p> <ol style="list-style-type: none"> 1. Viene mostrato un avviso all'utente 2. Il sistema mostra nuovamente la schermata di accesso al sistema
Requisiti non funzionali	La password digitata non deve essere visibile in maniera esplicita sulla schermata: sicurezza delle informazioni. Facilità di navigazione delle schermate
Punti aperti	

Titolo	GestioneUtenti
Descrizione	Gestione degli utenti

Attori	Amministratore
Relazioni	RegistrazioneUtente, EliminazioneUtente, CercaUtente, Autenticazione
Precondizioni	
Postcondizioni	
Scenario principale	<ol style="list-style-type: none"> 1. Autenticazione 2. L'Amministratore preme il pulsante per aprire la finestra di GestioneUtenti 3. Il sistema mostra all'Amministratore la schermata relativa alla GestioneUtenti 4. Il sistema mostra all'Amministratore l'elenco degli Utenti registrati alla stazione locale. 5. L'Amministratore può scegliere se filtrare gli Utenti visualizzati utilizzando CercaUtente 6. L'Amministratore può quindi decidere di effettuare una di queste due operazioni: <ol style="list-style-type: none"> a. RegistrazioneUtente b. EliminazioneUtente
Scenari alternativi	<p>Scenario a: Le credenziali fornite in fase di Autenticazione non risultano di grado Amministratore.</p> <ol style="list-style-type: none"> 1. Il sistema notifica l'utente che non ha sufficienti permessi per operare in questa sezione. 2. All'Utente viene mostrata la schermata principale <p>Scenario b: Non ci sono Utenti registrati alla stazione.</p> <ol style="list-style-type: none"> 1. Il sistema mostra un avviso all'Amministratore
Requisiti non funzionali	Integrità e sicurezza dei dati, facilità di navigazione delle schermate
Punti aperti	

Titolo	RegistrazioneUtente
Descrizione	L'Amministratore registra un Utente nel sistema
Attori	Amministratore, Utente
Relazioni	GestioneUtenti
Precondizioni	L'Utente non è registrato e l'Amministratore ha effettuato l'accesso presso il sistema.
Postcondizioni	L'Utente risulta registrato presso il sistema.
Scenario	<ol style="list-style-type: none"> 1. Il sistema mostra all'Amministratore una schermata contenente i

principale	<p>campi relativi alla registrazione dell'utente: username, password, nome, cognome, email.</p> <ol style="list-style-type: none"> 2. L'Amministratore cede la tastiera all'Utente per inserire i propri dati. 3. L'Utente inserisce i propri dati. 4. L'Amministratore verifica i dati e conferma l'operazione di registrazione. 5. Il sistema memorizza le informazioni relative all'utente 6. A video viene mostrato un messaggio di successo 7. Viene mostrata la schermata GestioneUtenti all'Amministratore
Scenari alternativi	<p>Scenario a: Utente già presente</p> <ol style="list-style-type: none"> 1. Viene segnalato un messaggio indicante la presenza dell'utente <p>Scenario b: I dati inseriti dall'Utente non sono validi</p> <ol style="list-style-type: none"> 1. L'Amministratore blocca l'operazione
Requisiti non funzionali	Integrità e sicurezza dei dati, facilità di navigazione delle schermate
Punti aperti	

Titolo	EliminazioneUtente
Descrizione	L'Amministratore elimina un utente dal sistema
Attori	Amministratore
Relazioni	GestioneUtenti
Precondizioni	L'Utente è presente e l'Amministratore ha effettuato l'accesso presso il sistema
Postcondizioni	L'Utente risulta eliminato dal sistema
Scenario principale	<ol style="list-style-type: none"> 1. L'Amministratore seleziona l'Utente da eliminare 2. Il sistema mostra una finestra per chiedere la conferma dell'operazione 3. L'Amministratore conferma l'operazione di eliminazione 4. Il sistema elimina l'utente 5. A video viene mostrata un messaggio di successo 6. Viene mostrata la schermata GestioneUtenti all'Amministratore
Scenari alternativi	<p>Scenario a: L'Amministrazione non conferma l'operazione</p> <ol style="list-style-type: none"> 1. Il sistema annulla l'operazione. 2. Il sistema mostra la schermata GestioneUtenti all'Amministratore
Requisiti non funzionali	Integrità e sicurezza dei dati, facilità di navigazione delle schermate
Punti aperti	

Titolo	CercaUtente
Descrizione	L'Amministratore filtra l'elenco di Utenti in base a dei parametri di ricerca
Attori	Amministratore
Relazioni	GestioneUtenti
Precondizioni	L'Amministratore ha effettuato l'accesso presso il sistema.
Postcondizioni	Il sistema ha mostrato all'Amministratore gli Utenti che rispettano i parametri di ricerca specificati.
Scenario principale	<ol style="list-style-type: none"> 1. Il sistema mostra all'Amministratore una schermata con i vari parametri di ricerca impostabili. 2. L'Amministratore specifica uno o più parametri di ricerca. 3. L'Amministratore clicca sul pulsante di ricerca. 4. Il sistema effettua la ricerca degli Utenti che rispettano i parametri specificati. 5. Il sistema mostra all'Amministratore gli Utenti trovati.
Scenari alternativi	
Requisiti non funzionali	Integrità e sicurezza dei dati, facilità di navigazione delle schermate. Rapidità di ricerca.
Punti aperti	

Titolo	GestioneStazioni
Descrizione	Gestione delle stazioni remote
Attori	Amministratore
Relazioni	RegistrazioneStazione, EliminazioneStazione, CercaStazione, Autenticazione
Precondizioni	
Postcondizioni	
Scenario principale	<ol style="list-style-type: none"> 1. Autenticazione 2. L'Amministratore preme il pulsante per aprire la finestra di GestioneStazioni 3. Il sistema fornisce la schermata per la gestione delle stazioni

	<ol style="list-style-type: none"> 4. Il sistema mostra l'elenco di tutte le stazioni remote registrate 5. L'Amministratore può filtrare le stazioni remote tramite CercaStazione 6. La schermata presenta un bottone per la RegistrazioneStazione e un altro per la EliminazioneStazione 7. L'Amministratore seleziona la scelta corrispondente <ol style="list-style-type: none"> a. RegistrazioneStazione b. EliminazioneStazione
Scenari alternativi	<p>Scenario a: Le credenziali fornite in fase di Autenticazione non risultano di grado Amministratore.</p> <ol style="list-style-type: none"> 1. Il sistema notifica l'utente che non ha sufficienti permessi per operare in questa sezione. 2. All'Utente viene mostrata la schermata principale <p>Scenario b: Non esistono stazioni remote registrate</p> <ol style="list-style-type: none"> 1. Il sistema mostra un avviso all'Amministratore
Requisiti non funzionali	Integrità e sicurezza dei dati, facilità di navigazione tra le schermate
Punti aperti	

Titolo	RegistrazioneStazione
Descrizione	L'Amministratore registra una stazione remota presso il sistema
Attori	Amministratore
Relazioni	GestioneStazioni
Precondizioni	La stazione non è registrata e l'Amministratore ha effettuato l'accesso presso il sistema
Postcondizioni	La stazione risulta registrata presso il sistema
Scenario principale	<ol style="list-style-type: none"> 1. Viene mostrata a video una schermata contenente i campi relativi alla registrazione della stazione: nome, locazione, indirizzo di rete 2. L'Amministratore inserisce i dati relativi alla stazione remota 3. Il sistema memorizza le informazioni relative alla stazione 4. A video viene mostrata un messaggio di successo 5. Viene mostrata la schermata di GestioneStazioni all'Amministratore
Scenari alternativi	<p>Scenario a: Stazione già presente</p> <ol style="list-style-type: none"> 1. Viene segnalato un messaggio di errore 2. Il sistema mostra la schermata di GestioneStazioni
Requisiti non funzionali	Integrità e sicurezza dei dati, facilità di navigazione delle schermate

Punti aperti	
---------------------	--

Titolo	EliminazioneStazione
Descrizione	L'Amministratore elimina una stazione remota dal sistema
Attori	Amministratore
Relazioni	GestioneStazioni
Precondizioni	La stazione è presente e l'Amministratore ha effettuato l'accesso presso il sistema
Postcondizioni	La stazione risulta eliminata dal sistema
Scenario principale	<ol style="list-style-type: none"> 1. L'Amministratore preme sul pulsante relativo all'eliminazione della stazione 2. Viene mostrato a video un messaggio di conferma dell'operazione contenente tutti i dati relativi alla stazione 3. Amministratore conferma l'operazione 4. Il sistema elimina le informazioni relative alla stazione 5. A video viene mostrata un messaggio di successo 6. Viene mostrata la schermata di GestioneStazioni all'Amministratore
Scenari alternativi	Scenario a: L'Amministratore non conferma l'operazione <ol style="list-style-type: none"> 1. Il sistema annulla l'operazione 2. Viene mostrata la schermata di GestioneStazioni all'Amministratore
Requisiti non funzionali	Integrità e sicurezza dei dati, facilità di navigazione delle schermate
Punti aperti	

Titolo	CercaStazione
Descrizione	L'Amministratore filtra l'elenco di stazioni remote in base a dei parametri di ricerca
Attori	Amministratore
Relazioni	GestioneStazioni
Precondizioni	L'Amministratore ha effettuato l'accesso presso il sistema
Postcondizioni	Il sistema ha mostrato all'Amministratore le Stazioni che rispettano i

	parametri di ricerca specificati
Scenario principale	<ol style="list-style-type: none"> 1. Il sistema mostra all'Amministratore una schermata con i vari parametri di ricerca impostabili 2. L'Amministratore specifica uno o più parametri di ricerca 3. L'Amministratore clicca sul pulsante di ricerca 4. Il sistema effettua la ricerca delle stazioni remote che rispettano i parametri specificati 5. Il sistema mostra all'Amministratore le stazioni remote trovate
Scenari alternativi	
Requisiti non funzionali	Integrità e sicurezza dei dati, facilità di navigazione delle schermate. Rapidità di ricerca.
Punti aperti	

Analisi del rischio

Valutazione dei beni

Bene	Valore	Esposizione
Sistema informativo	Alto Supporto alla registrazione dei dati rilevati ed alla visualizzazione degli stessi. Ausilio ad eventuali proiezioni statistiche. Critico dal punto di vista della sicurezza	Alta Perdita economica e d'immagine; costi di ripristino sistema
Sensore	Alto Rilevazione delle onde sismiche	Alto Mancata rilevazione di dati più o meno significativi
Record evento sismico	Medio Informazioni riguardanti uno specifico evento sismico	Media Perdita dei dati di un evento sismico più o meno significativo
Informazioni relative agli utenti	Medio Informazioni relative agli utenti del sistema, comprese le loro credenziali che permettono di accedere ai dati	Media Perdita d'immagine se gli utenti vengono compromessi
Informazioni relative agli amministratori	Alto Informazioni relative agli amministratori, comprese le loro credenziali che permettono di accedere e modificare parti critiche del sistema	Alta Perdita d'immagine: un attaccante con permessi amministrativi può compromettere seriamente il sistema

Analisi minacce e controlli

Minaccia	Probabilità	Controllo	Fattibilità
Furto d'identità (utente)	Alta	Log delle operazioni con etichetta utente	Basso costo e trasparente
Furto d'identità (amministratore)	Media	Accesso amministratore solo locale e log delle operazioni	Basso costo, limita i punti di attacco
Alterazione dei dati	Bassa	Uso di canale sicuro (SSL)	Costo medio, assicura

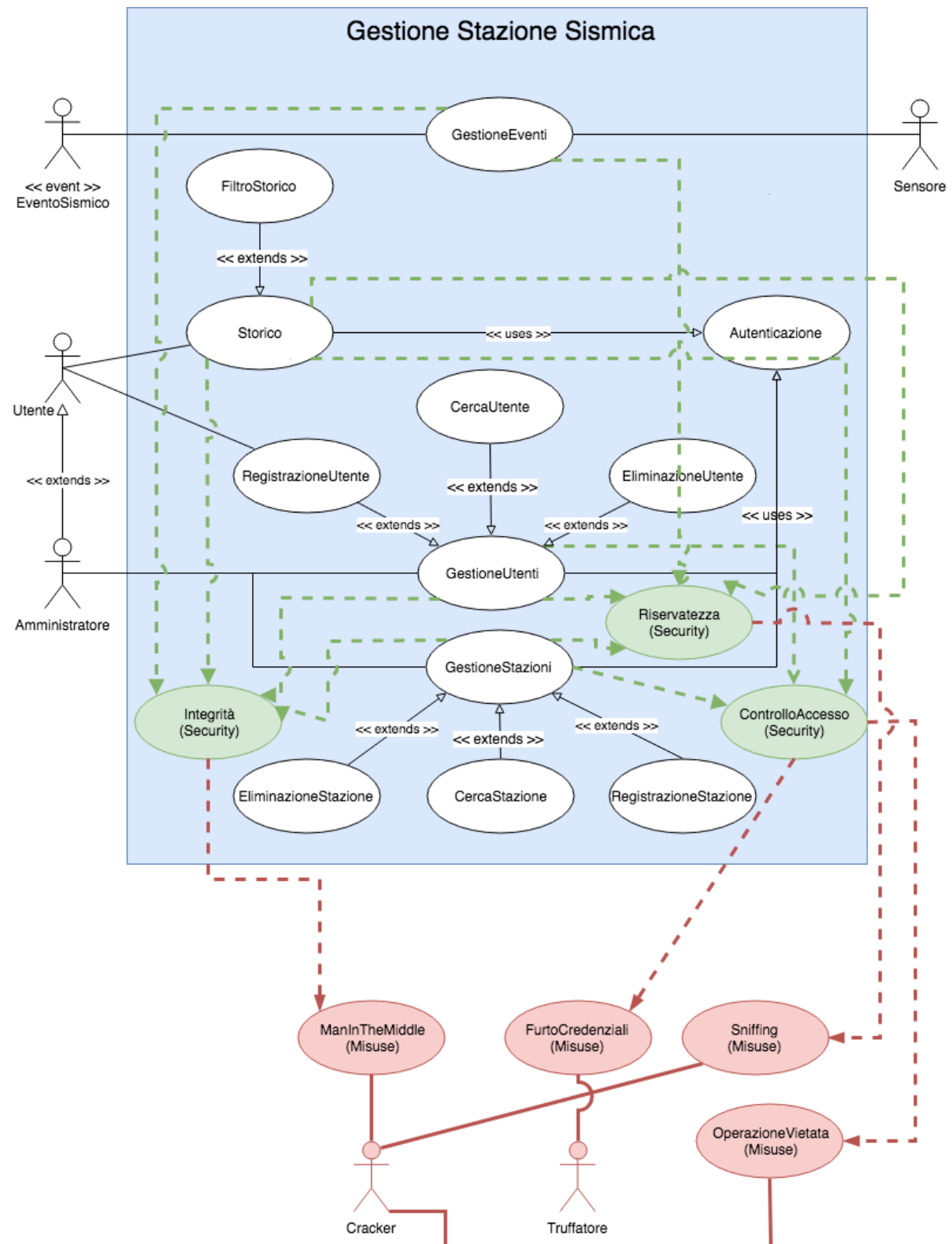
proveniente da stazione remota		e crittografia dei dati	l'integrità di dati sensibili
DoS	Media	Controllo e limitazione degli accessi	Costo basso. Impossibile prevenire questo tipo di attacco
Man in the middle	Media	Autenticazione end-to-end tramite certificati	Costo basso in aggiunta all'implementazione SSL
Manomissione del sensore	Bassa	Protezione del sensore fisico da accessi non autorizzati	Costo alto, bisogna proteggere l'edificio da accessi esterni

Analisi della tecnologia dal punto di vista della sicurezza

Tecnologia	Vulnerabilità
Architettura Client/Server	<ul style="list-style-type: none"> • Attacco Man in the Middle • Intercettazione (sniffing) delle comunicazioni • Attacco DoS
Autenticazione tramite credenziali	<ul style="list-style-type: none"> • L'utente rivela le sue credenziali in maniera volontaria • Vengono sottratte le credenziali all'utente tramite metodi di phishing. • L'utente sceglie una password facile da indovinare
Cifratura delle comunicazioni	<p>Per ottenere una comunicazione sicura, bisogna utilizzare sia una cifratura simmetrica che asimmetrica. Ognuna presenta delle vulnerabilità:</p> <p>Cifratura Simmetrica:</p> <ul style="list-style-type: none"> • Lunghezza della chiave: utilizzando chiavi corte, l'attaccante può trovarla con un approccio forza bruta. • Memorizzazione della chiave: la chiave deve essere memorizzata in una maniera sicura, altrimenti può facilmente essere ottenuta. • Tempo di vita della chiave: cifrando molte informazioni con la stessa chiave, l'attaccante ha più materiale per l'analisi del testo. <p>Cifratura Asimmetrica:</p> <ul style="list-style-type: none"> • Memorizzazione chiave privata: la

	<p>chiave privata di ogni stazione deve essere conservata in maniera sicura. Se un attaccante ottiene la chiave, può impersonare una stazione remota.</p> <ul style="list-style-type: none">● Falsificazione della chiave pubblica: un attaccante può, tramite attacco man in the middle, inoltrare una chiave pubblica falsificata all'utente finale. (Questo perché in questo progetto si è scelto di non utilizzare una Certificate Authority).● Lunghezza della chiave: Una chiave corta può essere facilmente decifrata dall'attaccante tramite forza bruta.
--	--

Security Use Case e Misuse Case



Caso d'uso: Integrità			
Percorso del caso d'uso: Integrità dei dati salvati dal sistema			
Rischi alla sicurezza: Un attaccante potrebbe corrompere o modificare dati relativi al sistema, come ad esempio lo storico degli eventi o gli account degli utenti			
Precondizioni: Il sistema ha memorizzato dati sensibili che non devono essere corrotti o cambiati da chi non autorizzato			
Interazioni dell'utente	Interazioni dell'attaccante	Requisiti del sistema	
		Interazioni del sistema	Azioni del sistema
	L'attaccante cerca di corrompere i dati del sistema		
			Il sistema dovrebbe impedire la manomissione dei dati sensibili
		Il sistema dovrebbe registrare il tentativo anomalo ed eventualmente informare l'amministratore	
Postcondizioni: Il sistema dovrebbe verificare che nessun dato sia stato corrotto			

Caso d'uso: Integrità			
Percorso del caso d'uso: Integrità nella trasmissione dei dati di un sensore			
Misuse case: ManInTheMiddle			
Rischi alla sicurezza: Un attaccante corrompe i dati sismici inviati da una stazione remota			
Precondizioni: <ol style="list-style-type: none"> 1. L'attaccante ha la possibilità di intercettare i dati di un sensore inviati da una stazione remota all'utente 2. L'attaccante ha la possibilità di modificare i dati intercettati 			

3. L'attaccante ha la possibilità di ritrasmettere i dati modificati all'utente			
Interazioni dell'utente	Interazioni dell'attaccante	Requisiti del sistema	
		Interazioni del sistema	Azioni del sistema
		La stazione remota dovrebbe inviare i dati del sensore all'utente	Il sistema dovrebbe impedire che i dati trasmessi siano modificati senza che l'utente se ne accorga
	L'attaccante intercetta i dati, li modifica e li inoltra all'utente		
L'utente riceve i dati corrotti			Il sistema dovrebbe rilevare la corruzione dei dati
		Il sistema dovrebbe notificare l'utente dell'avvenuta corruzione ed invalidare il sensore remoto	
Postcondizioni: Il sistema dovrebbe aver notificato l'utente della corruzione dei dati ed aver invalidato il sensore remoto			

Caso d'uso: Riservatezza			
Percorso del caso d'uso: Riservatezza della trasmissione di dati dalle stazioni remote			
Misuse case: Sniffing			
Rischi alla sicurezza: Un attaccante intercetta e visualizza dati inviati da una stazione remota ad un utente			
Precondizioni: L'attaccante ha la possibilità di intercettare i dati da una stazione remota ad un utente			
Interazioni dell'utente	Interazioni dell'attaccante	Requisiti del sistema	
		Interazioni del	Azioni del sistema

		sistema	
			Il sistema dovrebbe rendere illeggibile i dati sismici mentre transitano
		La stazione remota invia i dati sismici all'utente	
	L'attaccante intercetta i dati inviati		
Postcondizioni: Il sistema dovrebbe aver inviato dati sismici in una forma che l'attaccante non può leggere			

Caso d'uso: Riservatezza			
Percorso del caso d'uso: Riservatezza dei dati memorizzati dal sistema			
Rischi alla sicurezza: Un attaccante accede ai dati memorizzati dal sistema e visualizza dati riservati (ad esempio account e storico degli eventi)			
Precondizioni: Il sistema memorizza dati sensibili (ad esempio account e storico degli eventi)			
Interazioni dell'utente	Interazioni dell'attaccante	Requisiti del sistema	
		Interazioni del sistema	Azioni del sistema
			Il sistema dovrebbe rendere illeggibili i dati del sistema a persone non autorizzate
	L'attaccante accede ai dati privati del sistema		
Postcondizioni: Il sistema dovrebbe aver salvato i dati in una forma illeggibile per l'attaccante			

Caso d'uso: ControlloAccesso			
Percorso del caso d'uso: Furto di identità e autenticazione			
Rischi alla sicurezza: L'attaccante ruba i dati di identificazione e autenticazione ad un utente			
Misuse case: FurtoCredenziali			
Precondizioni: <ol style="list-style-type: none"> 1. L'attaccante non ha i dati di identificazione di un utente 2. L'attaccante non ha i dati di autenticazione di un utente 3. L'attaccante ha la possibilità di intercettare i tentativi di accesso di un utente 			
Interazioni dell'utente	Interazioni dell'attaccante	Requisiti del sistema	
		Interazioni del sistema	Azioni del sistema
		Il sistema dovrebbe inizialmente richiedere l'identità e l'autenticazione ad un utente	
L'utente si identifica ed autentica	L'attaccante intercetta i dati di accesso e tenta di rubare l'identità dell'utente		Il sistema dovrebbe proteggere i dati di un utente durante l'interazione, rendendoli illeggibili agli utenti esterni
			Il sistema dovrebbe identificare e autenticare l'utente
		Il sistema dovrebbe dare la possibilità all'utente autenticato di accedere alle sue risorse protette	
Postcondizioni: <ol style="list-style-type: none"> 1. Il sistema dovrebbe aver impedito all'attaccante di leggere i dati di accesso 2. Il sistema dovrebbe aver impedito all'attaccante di autenticarsi 3. Il sistema dovrebbe aver impedito all'attaccante di accedere a risorse protette 4. Il sistema dovrebbe aver registrato il tentativo fallito di autenticazione 			

Caso d'uso: ControlloAccesso		
Percorso del caso d'uso: Modifica di dati sensibili da parte di utenti senza permessi amministrativi		
Rischi alla sicurezza: Un utente regolare, senza permessi amministrativi, modifica parti critiche del sistema (come altri account o i sensori)		
Misuse case: OperazioneVietata		
Precondizioni: <ol style="list-style-type: none"> 1. L'utente ha dei dati di accesso validi 2. L'utente <i>non</i> ha privilegi amministrativi 		
Interazioni dell'utente (attaccante)	Requisiti del sistema	
	Interazioni del sistema	Azioni del sistema
L'utente tenta di modificare parti critiche del sistema.		
		Il sistema dovrebbe impedire all'utente di accedere a sezioni protette senza gli opportuni permessi.
	Il sistema notifica all'utente la mancanza dei permessi necessari.	
Postcondizioni: Il sistema dovrebbe aver impedito all'utente senza permessi amministrativi di aver modificato parti critiche		

Requisiti di Protezione dei Dati

Dall'analisi del rischio sono emersi altri requisiti riguardanti la protezione dei dati:

1. Creazione di un **sistema di log** per tracciare tutte le azioni avvenute sul sistema. Ogni interazione utente viene tracciata e registrata in maniera persistente. L'amministratore di una stazione è l'unico in grado di visionare i log. La lettura del file di log verrà effettuata attraverso un editor di testo esterno e non è prevista alcuna implementazione grafica all'interno del progetto.
2. I dati memorizzati dal sistema devono essere protetti da un eventuale attaccante con accesso al sistema, eventualmente adottando una cifratura dei dati.
3. I dati sismici trasmessi in remoto devono essere protetti da attacchi di tipo man in the middle, eventualmente adottando una cifratura dei dati in transito.

Descrizioni delle interfacce grafiche

Struttura

Il sistema presenta diverse sezioni raggiungibili da una barra di navigazione laterale. La struttura è indipendente dal tipo di account con cui è stato effettuato l'accesso; nel caso di utente attivo con bassi privilegi alcune funzionalità saranno semplicemente non visibili o bloccate con un opportuno messaggio informativo.

Schermata di accesso

All'avvio (o dopo il logout) l'intero sistema è protetto da una schermata di accesso. La business logic si occuperà di verificare se i dati inseriti corrispondano a quelli di un account registrato in modo persistente. Se questi sono corretti, verrà effettuato l'accesso, altrimenti sarà mostrato un messaggio di errore. Una schermata simile è disponibile per l'accesso remoto, in tal caso la business logic sarà quella di competenza della stazione remota.



Sismio

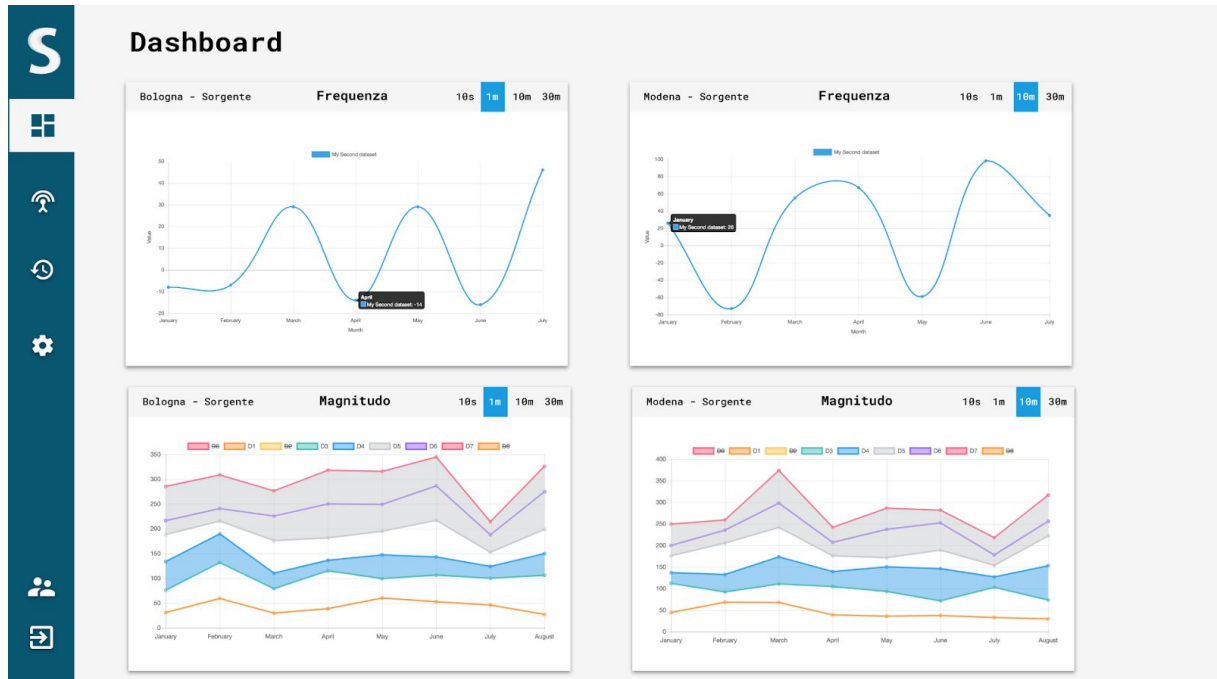
Username

Password

ACCEDI

Dashboard

Questa è la sezione predefinita ad accesso effettuato. Essa raggruppa le principali informazioni elaborate da diverse sorgenti contemporaneamente. La struttura a schede permette una rapida e significativa analisi a “colpo d’occhio”. Qui verranno renderizzati i grafici relativi all’elaborazione dei dati.



Gestione stazioni remote

Tramite questa sezione, visibile solo all'amministratore, è possibile effettuare tutte le operazioni volte alla gestione delle stazioni remote. Una vista a tabella mostra i dati reperiti dal sistema di persistenza. E' possibile effettuare operazioni di creazione e ricerca, oltre a l'eliminazione per ogni stazione (riga della tabella).

Gestione stazioni remote				
		Associa nuova		Cerca qui
Nome	Stazione d'origine	IP : porta	Chiave pubblica	
BS01	Bologna	10.5.23.56:4567	abjduhg3jJAHbd...	
MS01	Modena	10.11.54.2:1235	JDb31sn3jJAHbd...	
RS3	Reggio Emilia	12.9.56.23:6729	Bsdo2sn3jJAHbd...	
CS4	Cesena	10.9.9.5:5423	B7kKo2sn3jJAHbd...	

Storico eventi

La vista a tabella mostra lo storico degli eventi reperiti da dati persistenti. Questa sezione è di sola lettura, nessuna operazione di modifica è permessa. Rilevante è la scelta dei colori per la priorità dei relativi eventi. Sono disponibili operazioni di ricerca e filtraggio.

Storico eventi				
		Cerca qui	Filtro eventi sismici	Priorità INFO
			Inizio 06-05-2018 14:00	Fine 08-05-2018 20:00
Data e ora	Tag	Stazione d'origine	Messaggio	Priorità
08-05-2018 19:56:18	FREQUENZA	Bologna - S1	Brevi picchi negli ultimi 15 m...	INFO
08-05-2018 14:56:18	MAGNITUDO	Modena - S1	Scossa breve magnitudo 2	ALERT
07-05-2018 11:56:18	FREQUENZA	Bologna - S1	Frequenti scosse nell'ultimo m...	WARNING
06-05-2018 14:56:18	MAGNITUDO	Modena - S1	Scossa magnitudo 5 prolungata ...	CRITICAL

Gestione account

Permette la lettura e l'eliminazione dei dati relativi agli account degli utenti registrati, con funzionalità di ricerca e filtraggio. Il campo "login remoto", se abilitato, consente all'account corrispondente di effettuare il login da una stazione diversa dall'attuale in remoto. Visibile solo all'amministratore.

S

☐

📶

🕒

⚙️

👤

📄

Gestione account

Aggiungi nuovo

Cerca qui 🔍

Email	Username	Tipo	Remote login	
a@example.it	a-user	user	<input type="checkbox"/>	<div>🗑️</div>
b@example.it	b-user	user	<input checked="" type="checkbox"/>	<div>🗑️</div>
admin@example.it	c-admin	admin	<input checked="" type="checkbox"/>	<div>🗑️</div>
d@example.it	d-user	user	<input type="checkbox"/>	<div>🗑️</div>

✎

Analisi del problema

Analisi del documento dei requisiti

Analisi delle funzionalità

Tabella Funzionalità

Funzionalità	Tipo	Grado Complessità
GestioneEventi	Gestione e memorizzazione dati, interazione con l'esterno	Complessa
GestioneStazioni	Gestione e memorizzazione dati, interazione con l'esterno	Complessa
GestioneUtenti	Gestione e memorizzazione dati	Complessa
Autenticazione	Gestione dati, interazione con l'esterno	Semplice
Storico	Gestione e memorizzazione dati	Semplice
ScritturaLog	Memorizzazione dati	Semplice

GestioneEventi: Tabella Informazioni/Flusso

Informazione	Tipo	Livello protezione / privacy	Input / Output	Vincoli
Dato sensore hardware	Semplice	Media	Input	
Evento sismico	Composto	Alta	Output	
Data	Semplice	Media	Input	
Ora	Semplice	Media	Input	
Stazione d'origine	Composto	Bassa	Input	

GestioneStazioni: Tabella Informazioni/Flusso

Informazione	Tipo	Livello protezione / privacy	Input / Output	Vincoli
Nome	Semplice	Bassa	Input	Non più di 20 caratteri
Luogo	Semplice	Bassa	Input	Non più di 100 caratteri
IP	Semplice	Media	Input	Deve essere un indirizzo IP valido
Porta	Semplice	Media	Input	Deve essere una porta valida (intero compreso tra 1024 e 65535)

GestioneUtenti: Tabella Informazioni/Flusso

Informazione	Tipo	Livello protezione / privacy	Input / Output	Vincoli
Nome	Semplice	Alta	Input	Non più di 100 caratteri
Cognome	Semplice	Alta	Input	Non più di 100 caratteri
Email	Semplice	Alta	Input	Deve essere un indirizzo email valido
Username	Semplice	Molto alta	Input	Non più di 100 caratteri
Password	Semplice	Molto alta	Input	Non più di 100 caratteri
Permessi utente	Semplice	Alta	Input	
Abilitazione login remoto	Semplice	Alta	Input	Valore booleano

Autenticazione: Tabella Informazioni/Flusso

Informazione	Tipo	Livello protezione / privacy	Input / Output	Vincoli
Username	Semplice	Molto alta	Input	Non più di 100 caratteri

Password	Semplice	Molto alta	Input	Non più di 100 caratteri
----------	----------	------------	-------	--------------------------

Storico: Tabella Informazioni/Flusso

Informazione	Tipo	Livello protezione / privacy	Input / Output	Vincoli
Evento sismico <i>composto da:</i>	Composto	Alta	Input	
Data	Semplice	Media	Input	
Ora	Semplice	Media	Input	
Stazione d'origine	Semplice	Media	Input	
Tag	Semplice	Media	Input	
Messaggio	Semplice	Media	Input	Non più di 200 caratteri
Priorità	Semplice	Media	Input	Una delle seguenti: INFO, WARNING, ALERT, CRITICAL, FATAL

ScritturaLog: Tabella Informazioni/Flusso

Informazione	Tipo	Livello protezione / privacy	Input / Output	Vincoli
Utente	Semplice	Alta	Input	
Azione	Semplice	Alta	Input	Non più di 200 caratteri
Ora	Semplice	Media	Input	
Data	Semplice	Media	Input	

Analisi dei vincoli

Tabella dei vincoli

Requisito	Categoria	Impatto	Funzionalità
Integrità dei dati	Integrità	Peggiora la velocità di scrittura e di trasmissione ma garantisce una migliore protezione dei dati e una qualità superiore di questi; nella trasmissione e nella memorizzazione essa permette una maggiore probabilità di esito positivo e prevenzione degli errori	GestioneEventi, RegistrazioneUtente, EliminazioneUtente, CercaUtente, GestioneUtenti, RegistrazioneStazione, EliminazioneStazione, CercaStazione, GestioneStazioni
Sicurezza dei dati	Sicurezza	Peggiora tempo di risposta, migliora la privacy dei dati memorizzati	RegistrazioneUtente, EliminazioneUtente, CercaUtente, GestioneUtenti, RegistrazioneStazione, EliminazioneStazione, CercaStazione, GestioneStazioni, Autenticazione
Controllo Accessi	Sicurezza	Peggiora tempo di risposta e usabilità, migliora la privacy dei dati	GestioneUtenti, GestioneStazioni, Storico
Facilità di navigazione delle schermate	Usabilità	Cercare di migliorare	Storico, FiltroStorico, Autenticazione, GestioneUtenti, RegistrazioneUtente, EliminazioneUtente, CercaUtente, GestioneStazioni, RegistrazioneStazione, EliminazioneStazione, CercaStazione

Rapidità ricerca	Tempo di risposta	Cercare di migliorare	FiltroStorico, CercaUtente, CercaStazione
Rapidità di lettura e scrittura dati	Tempo di risposta	Cercare di migliorare	GestioneEventi
Efficienza analisi	Efficienza	Migliora la qualità e l'affidabilità del risultato	GestioneEventi
Tempo di risposta	Tempo di inoltramento	Cercare di migliorare	GestioneEventi

Analisi delle interazioni

Tabella maschere

Maschera	Informazioni	Funzionalità
Home Dashboard (schermata principale)	Grafici in tempo reale contenenti i risultati dell'analisi, notifiche di eventi, possibilità di navigazione verso le altre schermate	GestioneEventi
View Autenticazione	Username, password	Autenticazione
Home Storico	Data, ora, tag, stazione d'origine, messaggio, priorità	Storico
View FiltroStorico	Barra di ricerca, menù a tendina contenente lista delle priorità degli eventi, input data e ora inizio, input data e ora fine	FiltroStorico
Home GestioneUtenti	Email, username, tipo, abilitazione login remoto	GestioneUtenti
View RegistrazioneUtente	Input per nome, cognome, email, username, password, tipo, abilitazione login remoto	RegistrazioneUtente
View EliminazioneUtente	Messaggio di conferma con nome, cognome, email dell'utente	EliminazioneUtente
View CercaUtente	Input di testo (barra di ricerca)	CercaUtente
Home GestioneStazioni	Nome, locazione, indirizzo di rete	GestioneStazioni
View RegistrazioneStazione	Input per nome, locazione, indirizzo di rete	RegistrazioneStazione
View EliminazioneStazione	Messaggio di conferma con nome, locazione, indirizzo di rete	EliminazioneStazione
View CercaStazione	Input di testo (barra di	CercaStazione

	ricerca)	
--	----------	--

Tabella sistemi esterni

Sistema	Descrizione	Protocollo di interazione	Livello di sicurezza
Sensore	Sistema hardware che invia un flusso di informazioni al computer relativo alle vibrazioni rilevate	Il sensore invia tramite porta seriale un flusso continuo di informazioni al computer. Il sistema è in ascolto su tale interfaccia per effettuare la lettura	Medio. Il sensore non è attaccabile informaticamente, ma fisicamente scollegabile

Analisi dei ruoli e delle responsabilità

Ruolo	Responsabilità	Maschere	Riservatezza	Numerosità
Utente	Visualizza i dati dei sensori in tempo reale e può visionare lo storico degli eventi sismici. In caso abbia abilitato l'accesso remoto ad una specifica stazione, può leggere i dati del sensore collegato.	Home Dashboard, View Autenticazione, Home Storico, View FiltroStorico,	E' richiesto un alto grado di riservatezza.	Il numero massimo di utenti è limitato unicamente dalle risorse del sistema.
Amministratore	Gestione di tutte le informazioni riguardo le figure direttamente coinvolte con il sistema: gestione degli utenti, potendo registrarli ed eliminarli, gestione delle stazioni, potendo registrarle ed eliminarle. Ha inoltre le funzionalità di un Utente.	Tutte	E' richiesto un grado di riservatezza molto alto.	E' sufficiente una persona per la gestione delle informazioni coinvolte.

Utente: Tabella Ruolo-Informazioni

Informazione	Tipo di Accesso
Storico	Lettura
Sensore	Lettura
Stazioni*	Lettura

* L'utente può leggere una lista di stazioni solo nel momento in cui aggiunge una nuova sorgente remota, ma non può accedere alla Home GestioneStazioni che invece è accessibile solo dall'amministratore.

Amministratore: Tabella Ruolo-Informazioni

Informazione	Tipo di Accesso
Storico	Lettura
Sensore	Lettura/Scrittura
Stazioni	Lettura/Scrittura
Utenti	Lettura/Scrittura
Log*	Lettura

* Come specificato precedentemente, l'applicazione non prevede una schermata apposita per vedere i log, ma è previsto che l'Amministratore li possa visionare tramite un editor di testo esterno.

Scomposizione del problema

Funzionalità	Scomposizione
GestioneUtenti	RegistrazioneUtente, EliminazioneUtente, CercaUtente
GestioneStazioni	RegistrazioneStazione, EliminazioneStazione, CercaStazione

GestioneUtenti: Tabella Sotto-Funzionalità

Sotto-funzionalità	Sotto-funzionalità	Legame	Informazioni
RegistrazioneUtente	EliminazioneUtente	Non si può eliminare un utente se questo non è registrato	Username

GestioneStazioni: Tabella Sotto-Funzionalità

Sotto-funzionalità	Sotto-funzionalità	Legame	Informazioni
RegistrazioneStazione	EliminazioneStazione	Non si può eliminare una stazione se questa non è registrata	Identificativo stazione

Creazione modello del dominio

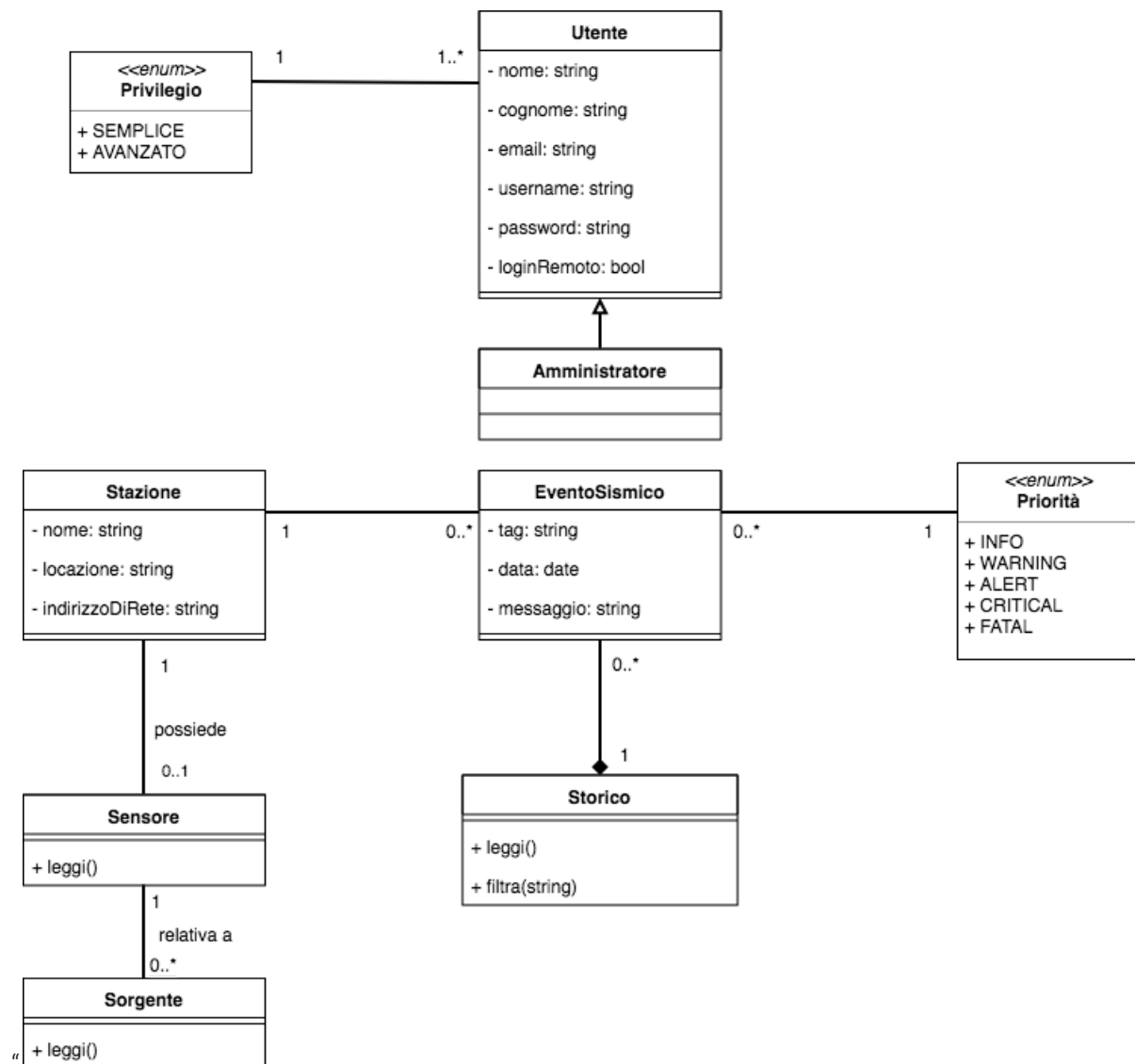
Privilegio:

- SEMPLICE: privilegio annesso all'utente
- AVANZATO: privilegio annesso all'amministratore

Priorità: gravità dell'evento sismico in ordine crescente

1. INFO
2. WARNING
3. ALERT
4. CRITICAL
5. FATAL

Si introduce il concetto di **Sorgente** che permette l'astrazione del sistema esterno Sensore, rendendo trasparente la sua locazione fisica (locale o remota). Il sistema mappa automaticamente la sorgente ad un sensore locale o remoto a seconda delle necessità, in maniera che l'applicativo possa analizzarne i dati indipendentemente dalla loro provenienza.



Architettura logica

Struttura

Diagramma dei package

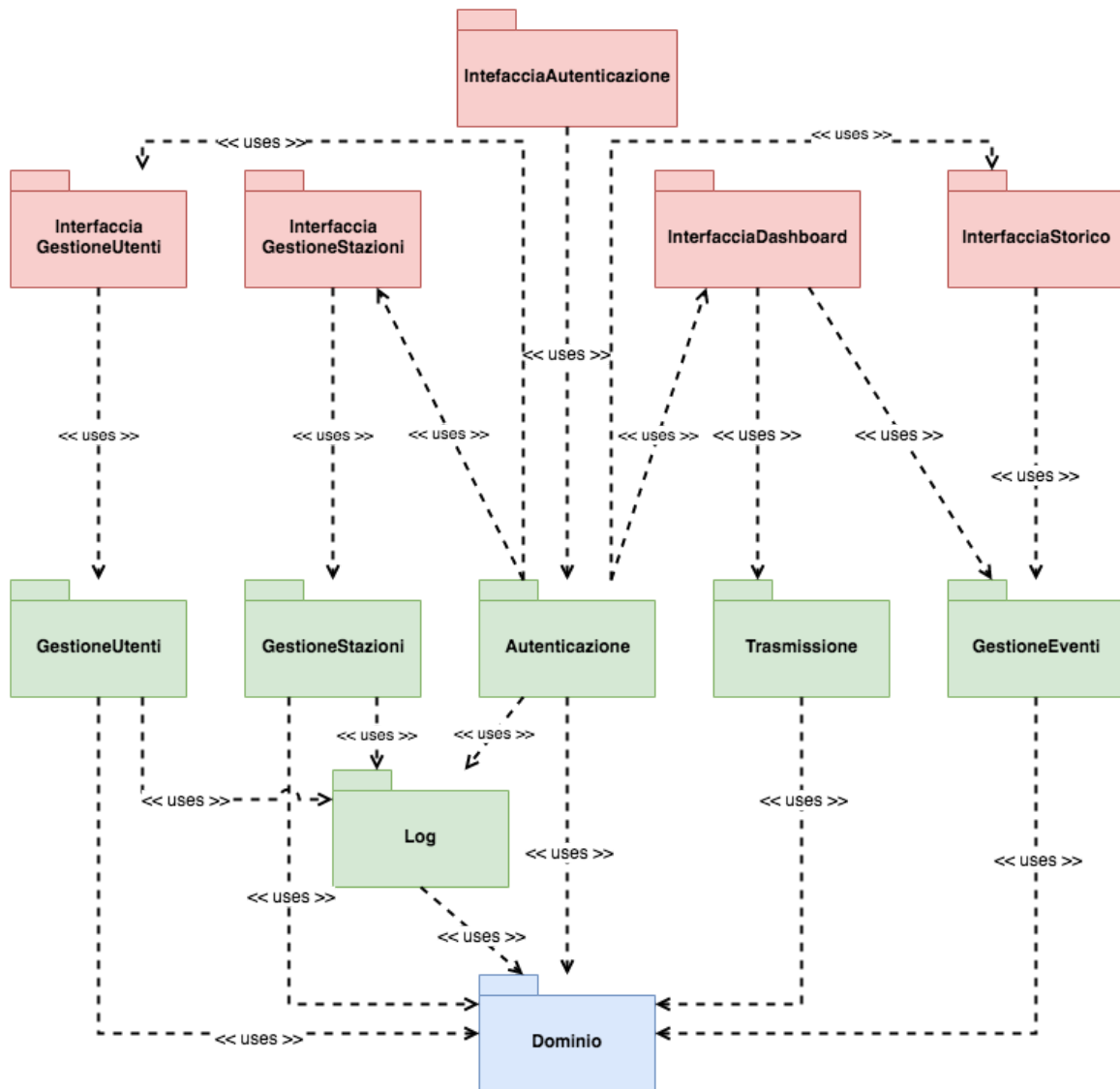
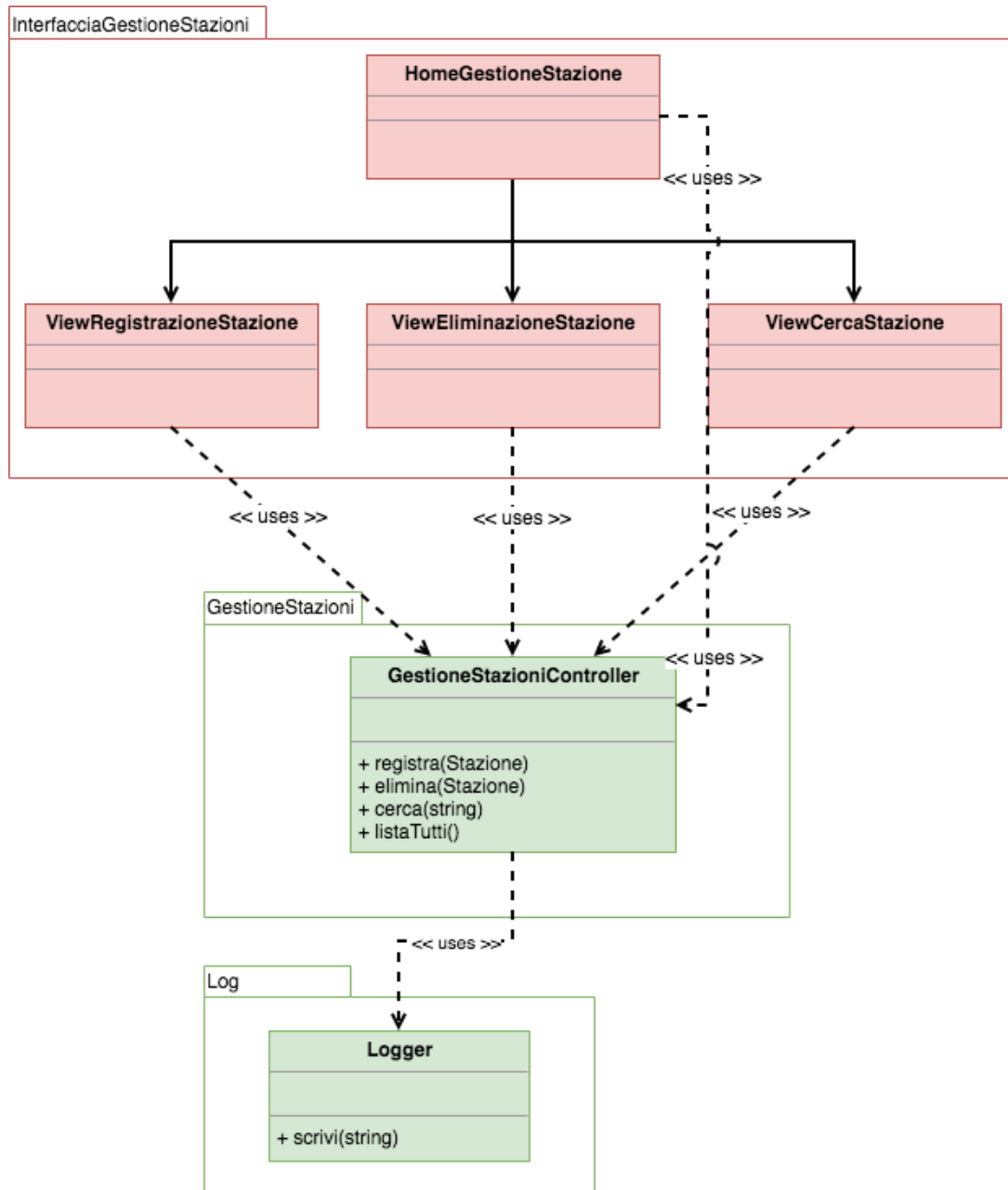


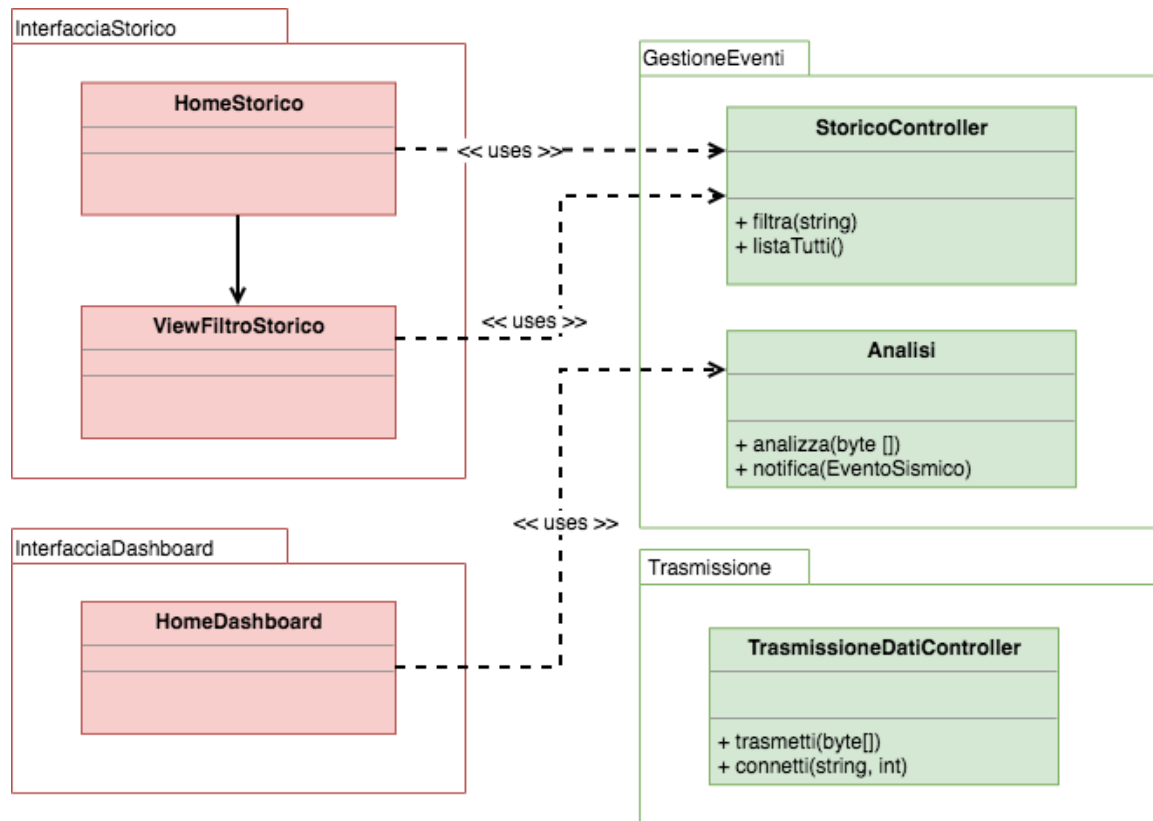
Diagramma delle classi



GestioneUtentiController è l'entità che si occupa di gestire gli utenti del sistema in maniera persistente. L'azione `verificaCredenziali()` è scaturita da `autentica()` di **AutenticazioneController** ed effettua la verifica delle credenziali di accesso appartenenti ad un determinato Utente.

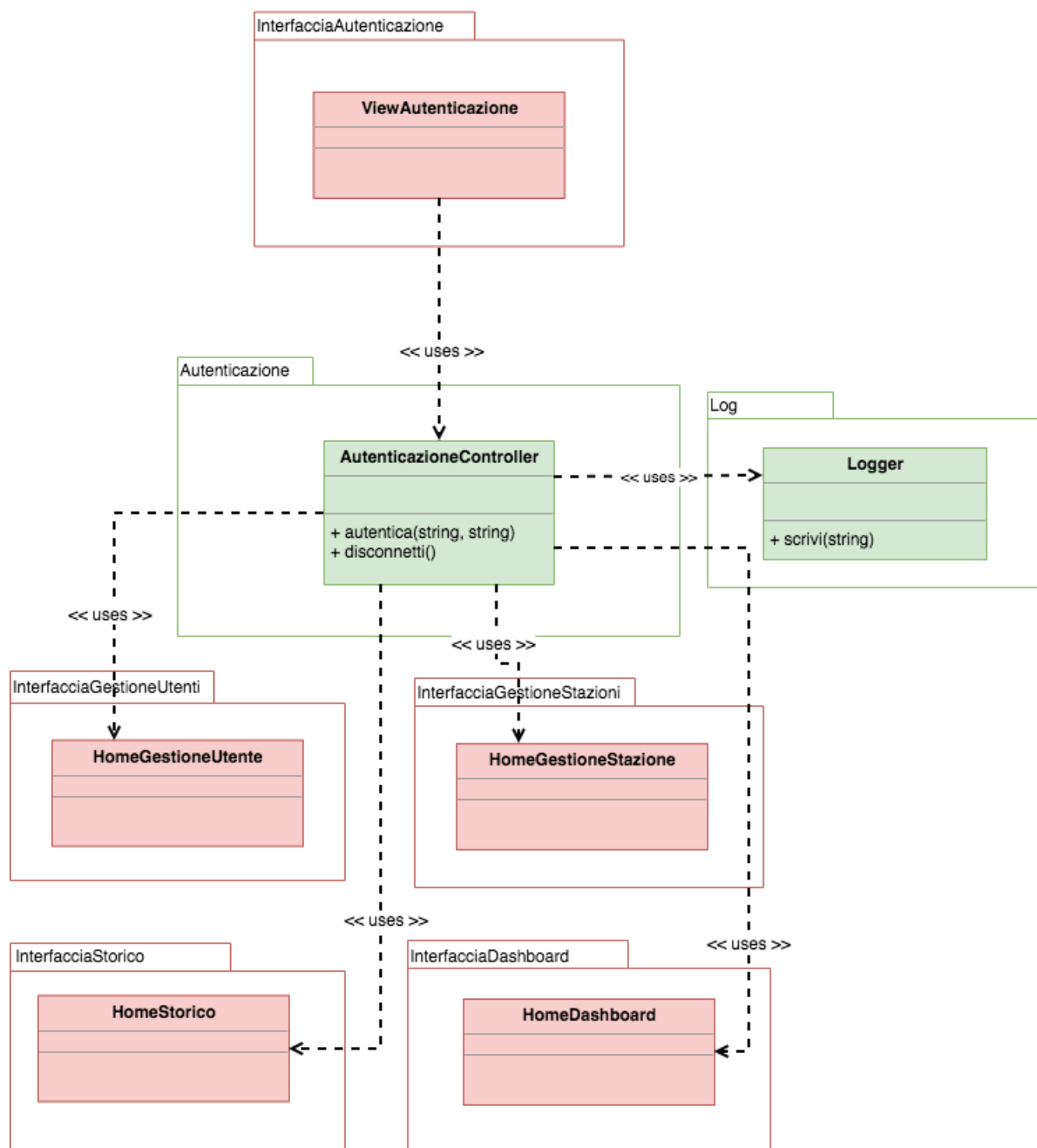


GestioneStazioniController è l'entità che si occupa di gestire le stazioni remote collegate al sistema in maniera persistente.



StoricoController si occupa di leggere e filtrare gli eventi sismici registrati in maniera persistente dal sistema.

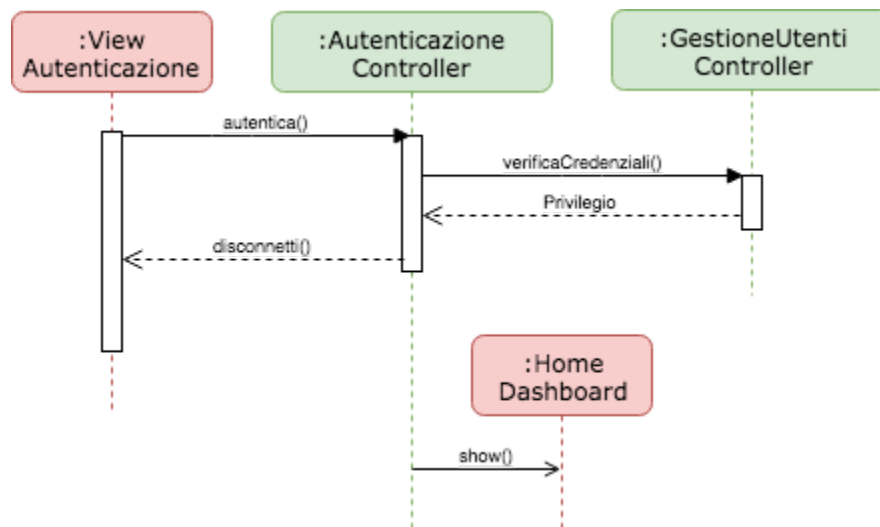
TrasmissioneDatiController si occupa di inviare i dati di un sensore ad una stazione remota. Viene eseguito indipendentemente dal resto dell'applicazione e non richiede alcuna interfaccia grafica.



AutenticazioneController gestisce la sessione di un utente all'interno del sistema. Interroga **GestioneUtentiController** per verificare se le credenziali sono corrette e si occupa di mantenere in memoria i dati di un utilizzatore per tutta la durata della sua sessione.

Interazione

Diagramma di sequenza: Autenticazione eseguita con successo



L'azione *disconnetti()* di **AutenticazioneController** rappresenta la procedura di logout di un utente. Terminata la procedura il sistema presenta **ViewAutenticazione** in attesa di un nuovo accesso.

Diagramma di sequenza: Lettura Sensore

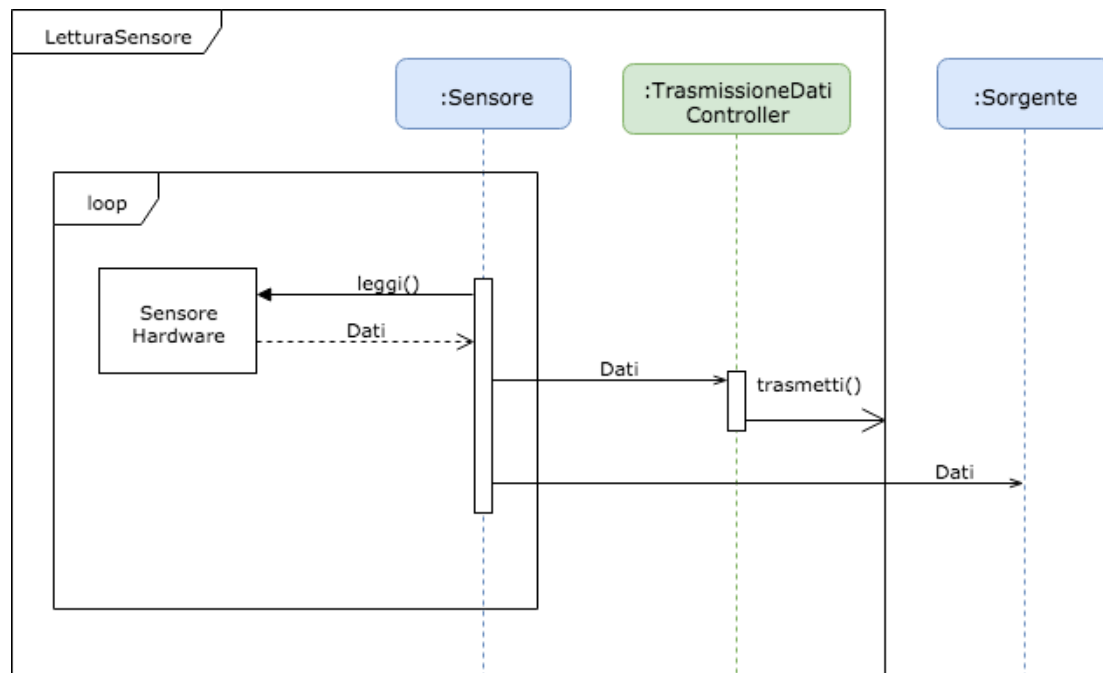


Diagramma di sequenza: Lettura Sorgente

La lettura della sorgente permette di rendere trasparente all'utente la fonte dei dati, indipendentemente dal fatto che questa sia locale o remota. Nello schema sotto riportato sono rappresentati i due casi, ovvero una lettura locale ed una remota.

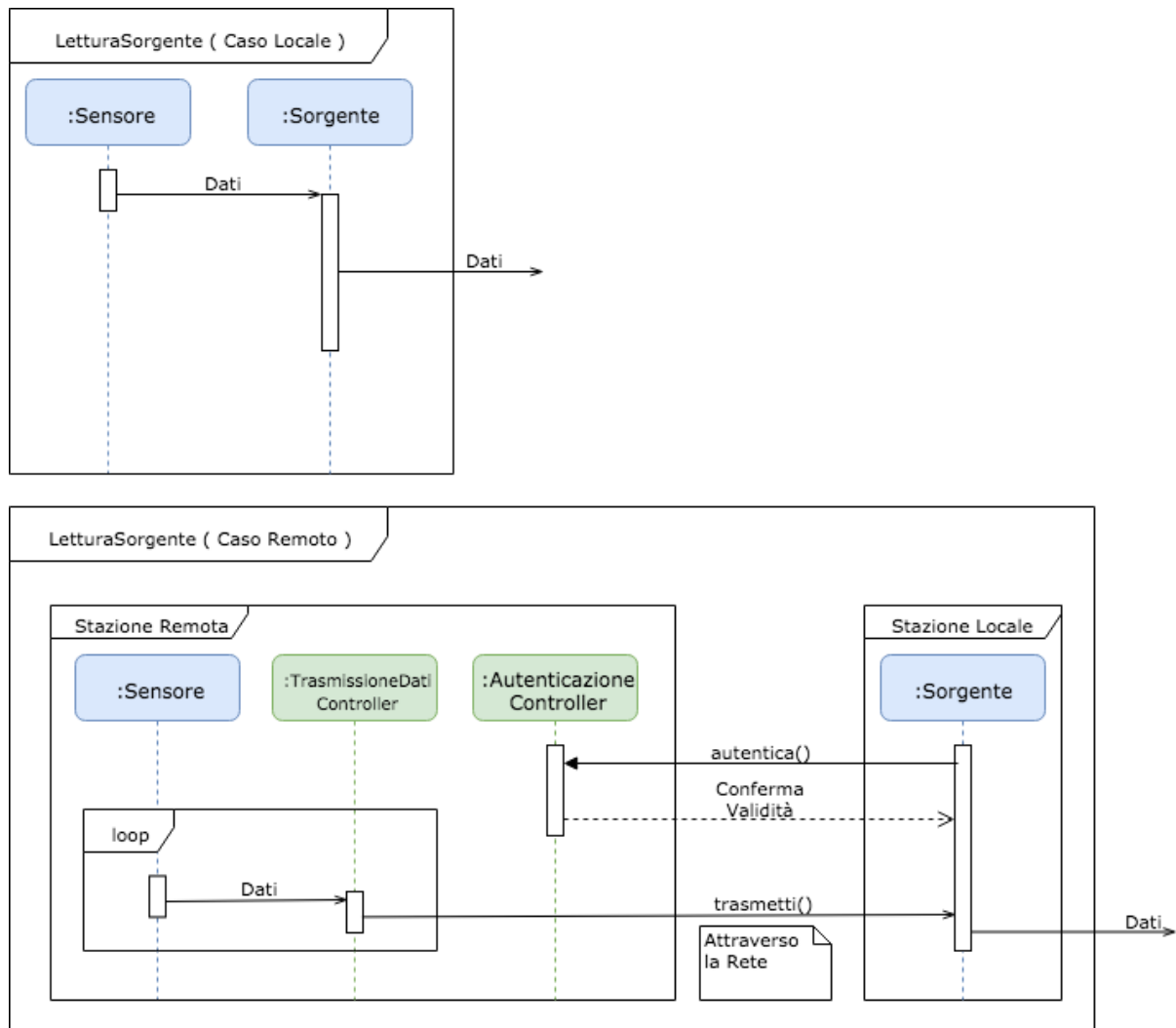
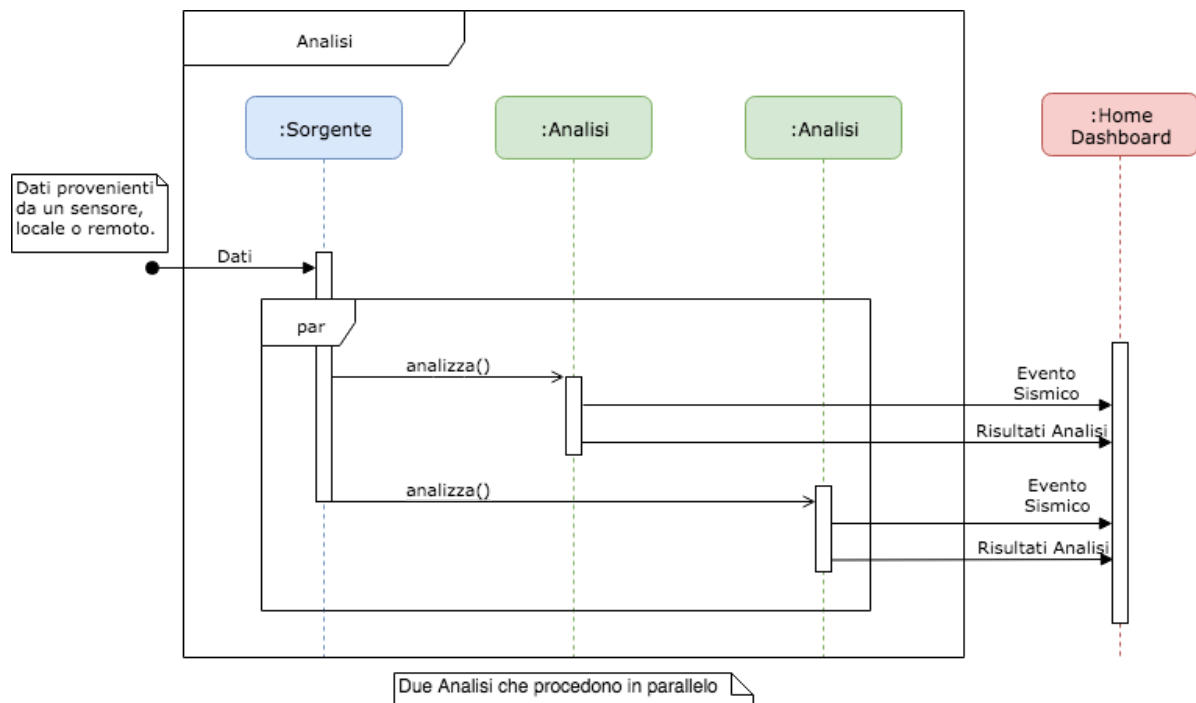


Diagramma di sequenza: Analisi



Le Analisi visualizzate nel grafico rappresentano due tipi differenti di elaborazioni, riguardanti Magnitudo e Frequenza. In futuro, il sistema permetterà di ampliare le tipologie di analisi disponibili.

Comportamento

Dopo un'attenta riflessione con la Prof.ssa Molesini, riteniamo che non esistano entità nel progetto che richiedono un diagramma di stato.

Piano del lavoro

Il lavoro di sviluppo del progetto è stato suddiviso tra i vari membri del team, come indicato nella tabella sottostante:

Package	Progetto	Sviluppo
Dominio	Terzi, Pellegrino, Bagnacani	Terzi, Pellegrino, Bagnacani
Log	Terzi, Pellegrino, Bagnacani	Bagnacani
GestioneUtenti	Terzi, Pellegrino, Bagnacani	Terzi
GestioneStazioni	Terzi, Pellegrino, Bagnacani	Bagnacani
Autenticazione	Terzi, Pellegrino, Bagnacani	Terzi
Trasmissione	Terzi, Pellegrino, Bagnacani	Terzi
GestioneEventi	Terzi, Pellegrino, Bagnacani	Bagnacani, Terzi, Pellegrino
InterfacciaAutenticazione	Terzi, Pellegrino, Bagnacani	Pellegrino
InterfacciaGestioneUtenti	Terzi, Pellegrino, Bagnacani	Pellegrino
InterfacciaGestioneStazioni	Terzi, Pellegrino, Bagnacani	Bagnacani
InterfacciaDashboard	Terzi, Pellegrino, Bagnacani	Pellegrino
InterfacciaStorico	Terzi, Pellegrino, Bagnacani	Pellegrino

Dopo un'attenta valutazione, i tempi di rilascio previsti sono i seguenti:

- Progettazione: entro 2 settimane dalla data odierna.
- Sviluppo dei vari moduli con annessi test unitari: entro 2 settimane dalla fine della progettazione.
- Integrazione e testing del sistema: entro una settimana dalla fine dello sviluppo.

Sviluppi futuri

Il committente ha richiesto che nei prossimi anni vengano aggiunti altri moduli di analisi più sofisticati. Si richiede quindi al team di progettazione di tenere conto di questi sviluppi futuri in maniera da rendere il sistema flessibile ad eventuali aggiunte.

Le notifiche testuali saranno implementate in versioni future, permettendo ad esempio l'invio di messaggi agli utenti registrati alla funzionalità.

Piano di collaudo

Per garantire il corretto funzionamento del sistema sono necessari una gamma di test unitari e di integrazione che permettono di verificare la correttezza delle singole parti. Ne vengono riportati solo un paio per non caricare eccessivamente il documento.

```
[TestFixture]
public class TestUtente
{
    private Utente _utente;

    [SetUp]
    public void UtenteSetup() {
        _utente = new Utente("Marty", "McFly",
            "marty.mcfly@fromthepast.com", "marty", "password", Privilegio.SEMPLICE,
            true);
    }

    [Test]
    public void TestGetterProprieta()
    {
        Assert.That(_utente.nome, Is.EqualTo("Marty"));
        Assert.That(_utente.cognome, Is.EqualTo("McFly"));
        Assert.That(_utente.email,
            Is.EqualTo("marty.mcfly@fromthepast.com"));
        Assert.That(_utente.username, Is.EqualTo("marty"));
        Assert.That(_utente.password, Is.EqualTo("password"));
        Assert.That(_utente.privilegio, Is.EqualTo(Privilegio.SEMPLICE));
        Assert.IsTrue(_utente.loginRemoto);
    }

    [Test]
    public void TestSetterProprieta()
    {
        Utente user = new Utente();
        user.nome = "Doctor";
        user.cognome = "Strange";
        user.email = "doctor.strange@marvel.com";
        user.username = "docstrange";
        user.password = "password";
        user.privilegio = Privilegio.AVANZATO;
        user.loginRemoto = true;

        Assert.That(user.nome, Is.EqualTo("Doctor"));
        Assert.That(user.cognome, Is.EqualTo("Strange"));
```

```

        Assert.That(user.email, Is.EqualTo("doctor.strange@marvel.com"));
        Assert.That(user.username, Is.EqualTo("docstrange"));
        Assert.That(user.password, Is.EqualTo("password"));
        Assert.That(user.privilegio, Is.EqualTo(Privilegio.AVANZATO));
        Assert.IsTrue(user.loginRemoto);
    }
}

```

```

[TestFixture]
public class TestEventoSismico
{
    private Stazione stazione;
    private EventoSismico evento;

    [SetUp]
    public void EventoSetUp() {
        stazione = new Mock<Stazione>();
        evento = new EventoSismico("MAGNITUDO", new DateTime(2018, 9, 10,
10, 1, 25), "Magnitudo rilevata di 8.2", Priorita.CRITICAL, stazione);
    }

    [Test]
    public void TestGetterProprieta()
    {
        Assert.That(evento.tag, Is.EqualTo("MAGNITUDO"));
        Assert.That(evento.data, Is.EqualTo(new DateTime(2018, 9, 10, 10,
1, 25)));
        Assert.That(evento.messaggio, Is.EqualTo("Magnitudo rilevata di
8.2"));
        Assert.That(evento.priorita, Is.EqualTo(Priorita.CRITICAL));
        Assert.That(evento.stazione, Is.EqualTo(stazione));
    }

    [Test]
    public void TestSetterProprieta()
    {
        EventoSismico newEvent = new EventoSismico();
        newEvent.tag = "FREQUENZA"
        newEvent.data = new DateTime(2018, 9, 10, 10, 1, 25);
        newEvent.messaggio = "Frequenza rilevata di 2 picchi / min";
        newEvent.priorita = Priorita.ALERT;
        newEvent.stazione = stazione;
    }
}

```

```
        Assert.That(newEvent.tag, Is.EqualTo("FREQUENZA"));
        Assert.That(newEvent.data, Is.EqualTo(new DateTime(2018, 9, 10,
10, 1, 25)));
        Assert.That(newEvent.messaggio, Is.EqualTo("Frequenza rilevata di
2 picchi / min"));
        Assert.That(newEvent.priorita, Is.EqualTo(Priorita.ALERT));
        Assert.That(newEvent.stazione, Is.EqualTo(stazione));
    }
}
```


Progetto

Progettazione architetturale

Requisiti non funzionali

Dall'analisi dei requisiti sono emersi alcuni vincoli non funzionali, tra cui:

- Integrità dei dati
- Sicurezza dei dati
- Controllo accessi
- Facilità d'utilizzo
- Rapidità ricerca
- Rapidità di lettura e scrittura dei dati
- Efficienza analisi
- Tempo di risposta

L'integrità e la sicurezza dei dati in transito sono estremamente importanti per il buon funzionamento del sistema. L'utilizzo di un protocollo TLS di comunicazione sicura permette di rispettare questi requisiti, al costo di aggiungere overhead alla trasmissione. Un'attenta valutazione ha permesso di determinare che, pur essendo necessario un flusso di dati in tempo reale, le moderne connessioni a banda larga permettono di soddisfare la velocità richiesta.

Anche il controllo degli accessi è un aspetto importante del sistema e permette di garantire che esso non venga manomesso da utenti non autorizzati. Pur peggiorando la facilità d'utilizzo del sistema, la sicurezza è da mettere al primo posto.

La rapidità di ricerca, lettura e scrittura dei dati coinvolge vari aspetti e parti del progetto. La parte di trasmissione è già stata trattata nei paragrafi precedenti, mentre un altro sottosistema riguarda lo storico degli eventi. Per garantire una buona performance viene introdotto un database *SQLite*, che permette di effettuare ricerche in velocità.

Per rispettare il vincolo di efficienza dell'analisi si utilizzano le capacità multi-core dei moderni processori, rendendo le varie elaborazioni parallele.

Il tempo di risposta è legato alla capacità del sistema di trasferire i dati da un nodo ad un altro. Si è ritenuto fosse ragionevole essere avvisati entro 5 secondi dal verificarsi di un evento sismico. In generale il sistema dovrà essere più veloce, ma visti i requisiti è accettabile assumere quel ritardo nel caso peggiore.

Scelta dell'architettura

Il sistema prevede un'architettura cliente/servitore a due livelli. L'applicazione contiene contemporaneamente le funzionalità del cliente e del servitore, rendendo ogni stazione (nodo) autocontenuta. Come pattern architetturale è stato scelto il Model View Controller (MVC).

Cliente

Un cliente può connettersi ad un servitore tramite una connessione sicura *TLS*. Un handshake è necessario per verificare l'autenticità del servitore, tramite certificato digitale.

L'autenticazione remota permette a un cliente di accedere ad un server disponendo delle credenziali di accesso residenti nel server.

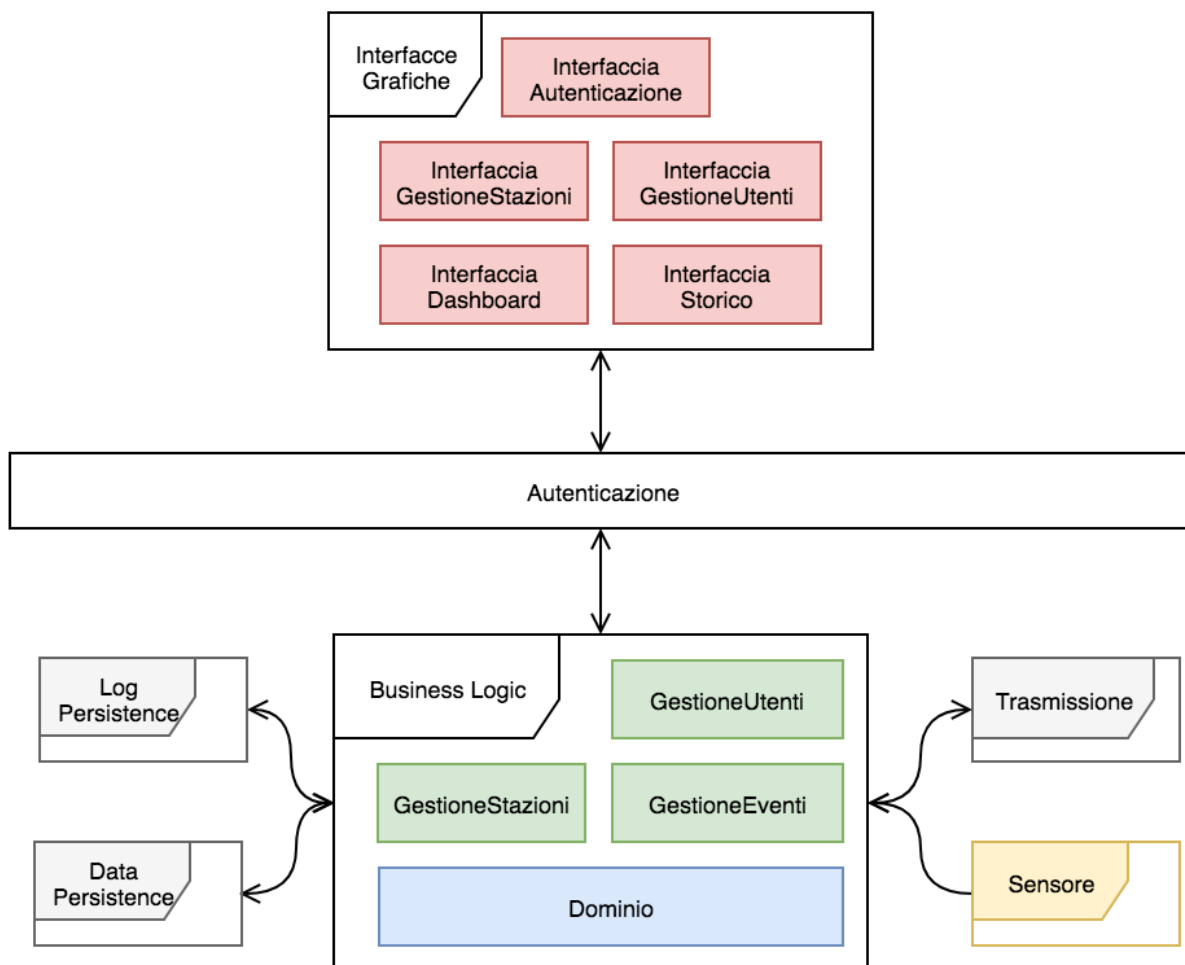
Servitore

Le funzionalità del server sono utilizzabili solo se la stazione possiede un sensore collegato e funzionante. All'avvio viene controllato lo stato del sensore, se risulta corretto viene esposto un servizio basato su *TCP/IP* al quale un cliente può connettersi.

Persistenza

Per assolvere alle necessità di persistenza del sistema, una gestione a soli file risulterebbe limitante. Allo stesso modo però, utilizzare un database completo ed indipendente risulterebbe inutilmente complesso. Si è perciò scelto di utilizzare *SQLite*, un database auto-contenuto molto leggero ed in grado di assolvere alle nostre necessità. Ogni stazione mantiene un sistema di persistenza unico e distinto dalle altre.

Viene riportata l'architettura del sistema:



Considerazioni sulla sicurezza relative alle tecnologie utilizzate

Dopo un'attenta analisi del sistema, si è notato che la maggior parte delle vulnerabilità tecnologiche possono essere ricondotte al **database** e la **trasmissione remota dei dati**.

Database

SQLite si basa su un unico file in cui vengono salvati tutti i dati e questo comporta una serie di possibili vulnerabilità tecniche:

Rischio	Accesso al file del database e lettura non autorizzata dei dati.
Descrizione	Essendo tutto il database contenuto in un normale file, come si può evitare che un attaccante possa leggerne il contenuto? Un qualunque utente della macchina in cui il software viene eseguito, con sufficienti permessi di lettura, è in grado di accedere ai dati.
Possibili soluzioni	Una possibile soluzione al problema è la cifratura dei dati stessi che, a seguito di un accesso non autorizzato, renderebbe illeggibile il contenuto a meno della conoscenza della chiave di decifratura. La libreria SQLite scelta offre nativamente la possibilità di cifrare i dati al suo interno.

Rischio	Accesso al file del database e corruzione dei dati.
Descrizione	Un attaccante con accesso in scrittura al computer potrebbe corrompere il file del database, rendendone impossibile l'utilizzo.
Possibili soluzioni	Una soluzione completa è difficile da realizzare. Il problema potrebbe tuttavia essere mitigato con regolari backup del database e controlli di integrità.

Trasmissione remota dei dati

Per rendere possibile una trasmissione remota dei dati sismici da una stazione ad un'altra, è necessario creare una connessione in tempo reale attraverso la rete. Questo canale deve garantire uno standard di qualità di servizio e sicurezza, in maniera tale che un attaccante non possa leggere o corrompere i dati in transito. Per fare ciò, si è scelto di utilizzare la classe SslStream, inclusa nel framework .NET. Questa permette di rendere sicura una connessione implementando il protocollo SSL TLS, fornendo quindi i meccanismi di identificazione e riservatezza necessari. Il corretto funzionamento si basa sull'uso di certificati, in questo caso autofirmati per evitare il costo di una Certificate Authority, che permettono di stabilire una connessione protetta.

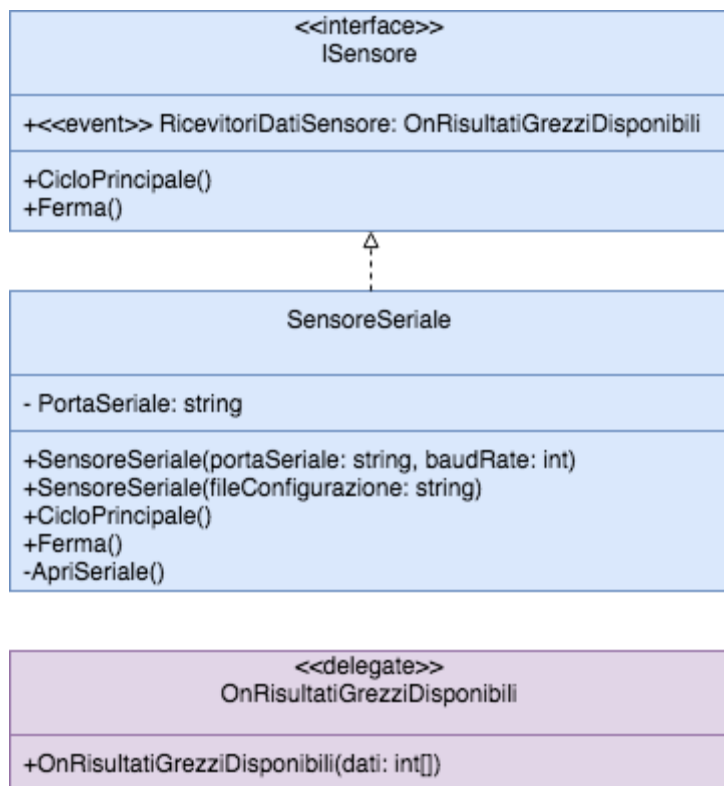
Rischio	Manomissione del certificato pubblico.
----------------	--

Descrizione	Un attaccante potrebbe, in fase di SSL handshake, sostituire il certificato pubblico della stazione remota con il suo, rendendo possibile un attacco <i>man in the middle</i> .
Possibili soluzioni	<p>Le possibilità sono molteplici:</p> <ul style="list-style-type: none"> ● Invece che usare un certificato auto-firmato, richiederne uno ad una Certificate Authority, che potrebbe poi essere usata per verificarne l'autenticità. ● In maniera analoga al popolare strumento unix <i>ssh</i>, mantenere un registro di chiavi pubbliche ammesse. All'atto di una connessione con una nuova stazione remota, l'amministratore dovrà verificare ed aggiungere manualmente il certificato al registro delle chiavi.

Progettazione di Dettaglio

Struttura

Package `io.sismio.sensore`



Questo package si occupa di definire l'astrazione e l'implementazione del concetto di sensore. Attraverso un ciclo indefinito principale è possibile leggere in maniera continuativa i valori dal sensore. Questi, sono rappresentati e contenuti dalla gestione dell'evento `OnRisultatiGrezziDisponibili`.

ISensore

Interfaccia che rappresenta il concetto di sensore. Definisce il suo comportamento tramite due azioni di avvio (`CicloPrincipale`) e di arresto (`Ferma`). Il resto del sistema usa questa interfaccia per non dipendere direttamente da un'implementazione specifica. Conseguenza l'introduzione di una proprietà `event` alla quale è possibile registrarsi per l'ascolto di nuovi dati disponibili, indipendentemente dal tipo di sensore.

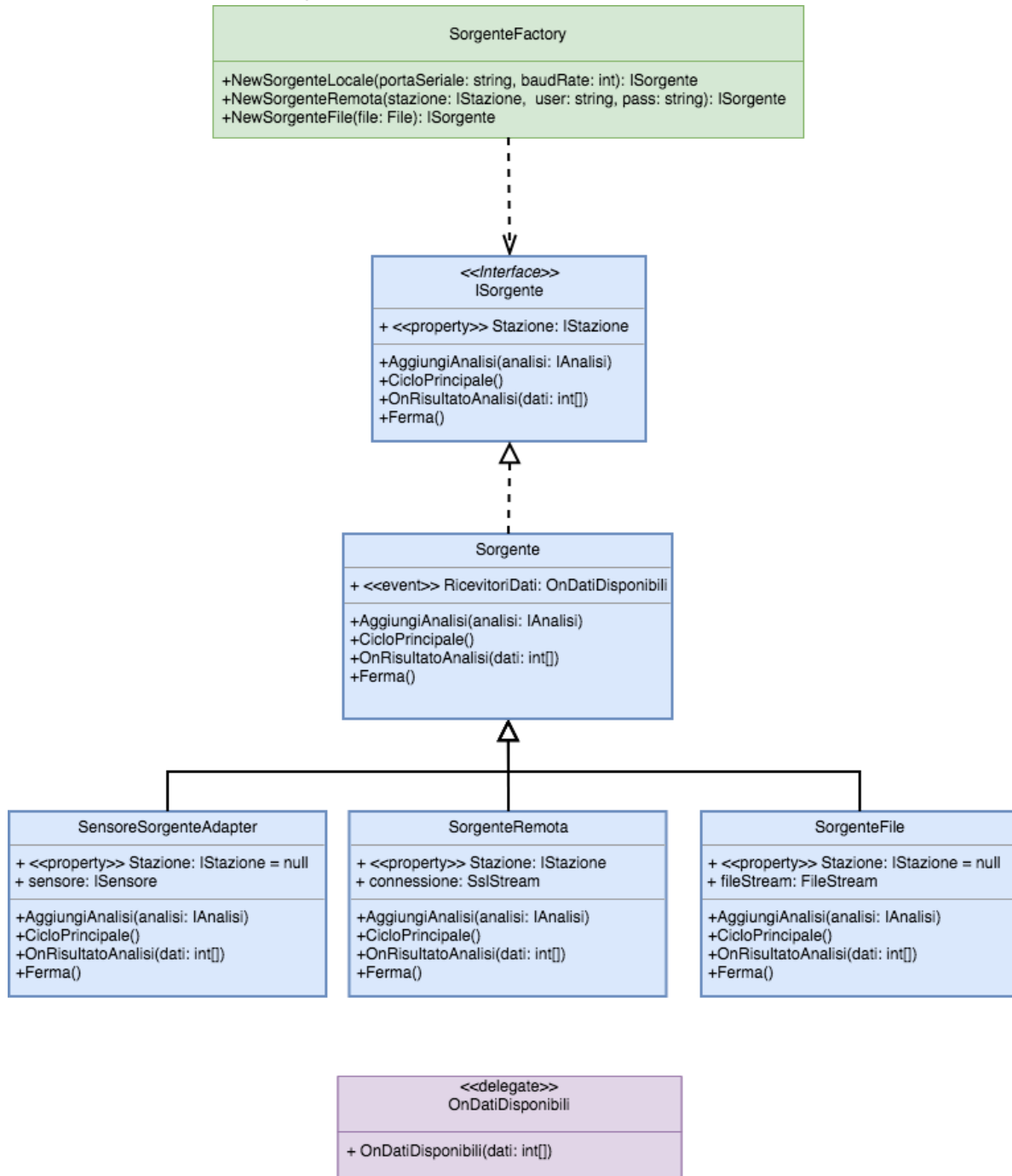
SensoreSeriale

Implementazione di **ISensore** che legge i dati da una porta seriale. Necessita del nome della porta (percorso) e della velocità di trasmissione (`baudRate`) espressa come intero. Si è deciso inoltre di fornire un costruttore aggiuntivo per esprimere queste informazioni in un file di configurazione, così da poter variare l'istanza di un **Sensore** senza dover modificare il codice dell'applicativo e rieseguire la sua compilazione.

OnRisultatiGrezziDisponibili (**Delegate**)

L'evento del delegato al momento di risultati disponibili verrà gestito; questo verrà utilizzato nell'implementazione dell'interfaccia ISensore. L'evento viene generato al riempimento di un buffer da parte dei dati forniti da ISensore. Questo buffer è posto come parametro del delegato affinché raggiunga gli ascoltatori.

Package `io.sismio.sorgente`



Questo package si occupa di rendere trasparente all'utente la fonte dei dati, ovvero il fatto che il sensore sia locale o remoto. A seguito sono indicate le principali classi che lo compongono, con la descrizione della loro funzionalità attesa.

SorgenteFactory

Il punto di accesso pubblico al package, realizza il pattern **Factory** e nasconde all'utente la creazione di istanze di tipo `ISorgente`. L'utilizzo di questo pattern è giustificato dal fatto che le implementazioni di `ISorgente` necessitano di configurazioni notevolmente differenti tra di loro, ma al contempo presentano alcune caratteristiche comuni. Si vuole dunque celare ogni ambiguità fornendo all'utilizzatore una factory che semplifica la richiesta di istanze scarnando all'essenziale le informazioni richieste dalle configurazioni.

ISorgente

Interfaccia pubblica che rappresenta il concetto di sorgente. Definisce il comportamento traendo spunto da quello di ISensore, racchiudendo azioni di avvio e arresto. Permette di registrare ascoltatori per i nuovi dati disponibili tramite la definizione di un delegato. Il metodo `AggiungiAnalisi` è essenziale per permettere al sistema di associare una o più elaborazione dati ad una determinata sorgente. Infine è definita una proprietà d'interfaccia per specificare che una sorgente è associata ad una stazione.

Sorgente

Le implementazioni di ISorgente hanno dei blocchi di codice in comune. Ad esempio `AggiungiAnalisi` è la medesima attraverso i diversi tipi di sorgente. Per questo motivo si è deciso di introdurre una classe astratta che implementa tutte le parti in comune. Le implementazioni concrete di ISorgente dovrebbero estendere questa classe. Inoltre mantiene un delegato per permettere l'utilizzo di un pattern **Observer** implementato dal package `io.sismio.analisi`.

SensoreSorgenteAdapter (**Adapter**)

Questa classe implementa ISorgente specificando il comportamento di un sensore presente nel sistema locale (direttamente collegato alla macchina). Dal momento in cui il sensore è rappresentato da ISensore, questa classe realizza a tutti gli effetti un pattern **Adapter**, in quanto maschera l'interfaccia per esporne un'altra. I comportamenti di ISensore sono riflessi negli omonimi metodi `Ferma` e `CicloPrincipale`. Questa classe è l'implementazione del concetto di trasparenza che si è voluto attribuire alla sorgente riguardo un sensore locale.

SorgenteRemota

Implementazione di una sorgente remota che trae informazioni da un sensore *non* locale, attraverso una connessione sicura creata dal package `io.sismio.trasmissione`. Rispetto alla precedente *non* è un pattern Adapter in quanto non vi è una conversione di interfaccia, ma invece una lettura di informazioni a seguito di una connessione via rete. Questa classe è l'implementazione del concetto di trasparenza che si è voluto attribuire alla sorgente riguardo un sensore remoto.

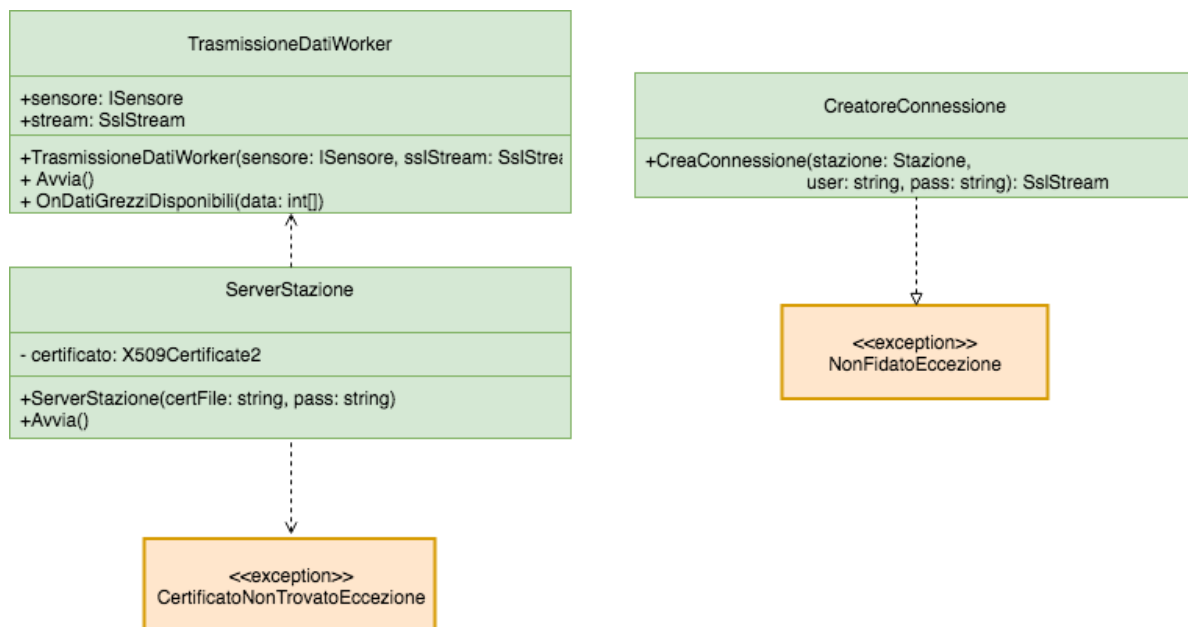
SorgenteFile

Implementazione che legge i valori di un sensore da un file locale esistente, di notevole utilità per test in fase di sviluppo.

OnDatiDisponibili (**Delegate**)

L'evento del delegato al momento di risultati disponibili verrà gestito; questo verrà utilizzato nell'implementazione dell'interfaccia ISorgente.

Package `io.sismio.trasmissione`



Questo package si occupa di rendere possibile la trasmissione remota dei dati di un sensore, curando anche gli aspetti legati alla sicurezza.

CreatoreConnessione

Si occupa di creare un canale sicuro di trasmissione utilizzando un oggetto `SslStream` reso disponibile dal framework .NET. Prende in ingresso un oggetto `Stazione` che rappresenta il nodo remoto a cui è fisicamente collegato il sensore da leggere. Inoltre riceve anche le credenziali che l'utente usa per autenticarsi alla stazione remota. Nel diagramma di sequenza "Creazione della connessione sicura", presente nella sezione di "Interazione", è spiegato il funzionamento di questo modulo.

TrasmissioneDatiWorker

Thread che si occupa di trasmettere i dati del sensore locale ad una stazione remota. Viene istanziato da `ServerStazione` ed è unico per ogni cliente. Continua ad inviare dati fino a quando la trasmissione non è interrotta o il sensore viene scollegato.

ServerStazione

Servitore che viene istanziato all'avvio del sistema e si pone in attesa di clienti. In particolare, a seguito di una richiesta di collegamento e dopo aver verificato opportunamente le credenziali del richiedente, istanzia `TrasmissioneDatiWorker` che invia i dati al cliente.

NonFidatoEccezione

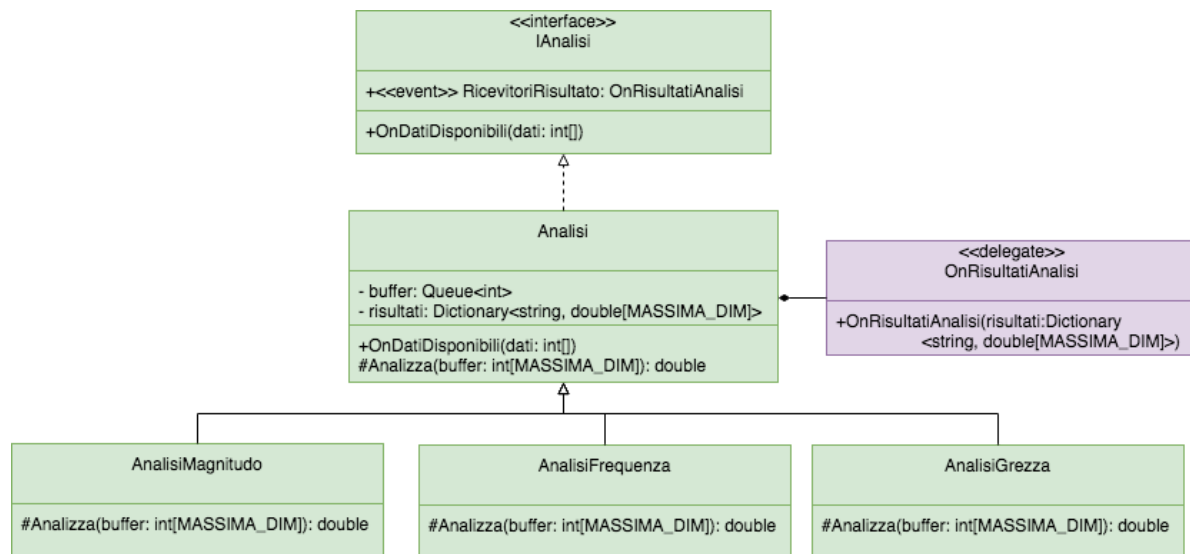
Eccezione che segnala una fonte non fidata. Viene lanciata dalla classe

`CreatoreConnessione` nel caso in cui la chiave pubblica della stazione remota non sia nell'elenco di quelle fidate.

CertificatoNonTrovatoEccezione

Eccezione che segnala la mancanza del certificato privato della stazione locale. Viene lanciata dalla classe `ServerStazione` all'avvio se la chiave privata non è stata trovata. Questo perché senza il certificato non si possono instaurare delle connessioni sicure.

Package `io.sismio.analisi`



Questo package si occupa degli aspetti legati all'elaborazione dei dati: riceve i valori da una Sorgente e, a seguito dell'analisi, genera gli eventi corrispondenti. Si occupa di notificare ed aggiornare i grafici, mediante il delegato `OnRisultatiAnalisi`. Ogni sorgente può avere più analisi associate, che vengono eseguite concorrentemente.

Le analisi vengono realizzate tramite il pattern **Strategy** per fornire un'alta flessibilità di elaborazione. In particolare, per definire un nuovo tipo di analisi basterà estendere la classe astratta `Analisi` e implementare la logica di calcolo nel metodo `Analizza()`.

`IAnalisi`

Interfaccia pubblica che rappresenta il concetto di analisi.

`Analisi`

Classe astratta che si occupa di implementare i metodi comuni a tutte le analisi, come la bufferizzazione dei dati. Mantiene un delegato per permettere il pattern **Observer**, utile alla View per ricevere i risultati dall'elaborazione. Nella classe `Analisi` è stata inserita una proprietà denominata "buffer"; essa verrà implementata attraverso l'utilizzo di una Queue.

Il suo scopo è quello di raccogliere ed accumulare i dati provenienti dal sensore per effettuare un'Analisi in blocco, permettendo di dare un miglior contesto ai valori letti e garantendo una migliore affidabilità del risultato.

`Analisi`, inoltre, convertirà la Queue in un array di interi, permettendo un'elaborazione più efficiente sotto vari aspetti. `MASSIMA_DIM` rappresenta la dimensione massima dell'array del blocco di dati che verranno analizzati; essa è impostata dagli sviluppatori su consiglio del progettista.

`AnalisiMagnitudo`

Questa analisi si occupa di analizzare il livello di magnitudo dei dati ricevuti, generando degli eventi di priorità variabile in base all'intensità.

AnalisiFrequenza

Monitora la quantità di scosse in un certo intervallo temporale, generando degli eventi di priorità variabile in base alla frequenza.

AnalisiGrezza

Restituisce i dati grezzi ricevuti senza ulteriori elaborazioni. La presenza di questa classe è motivata dal requisito del committente riguardante la lettura in tempo reale dei dati del sensore. Si è ritenuto che fosse una buona strategia quella di riutilizzare il meccanismo dell'Analisi per portare i dati del sensore ai grafici senza modifiche.

OnRisultatiAnalisi

Delegate che rappresenta l'evento di nuovi risultati disponibili a seguito dell'elaborazione dell'analisi. Viene utilizzato dalle analisi per notificare la View della presenza di nuovi dati disponibili.

Package `io.sismio.database`

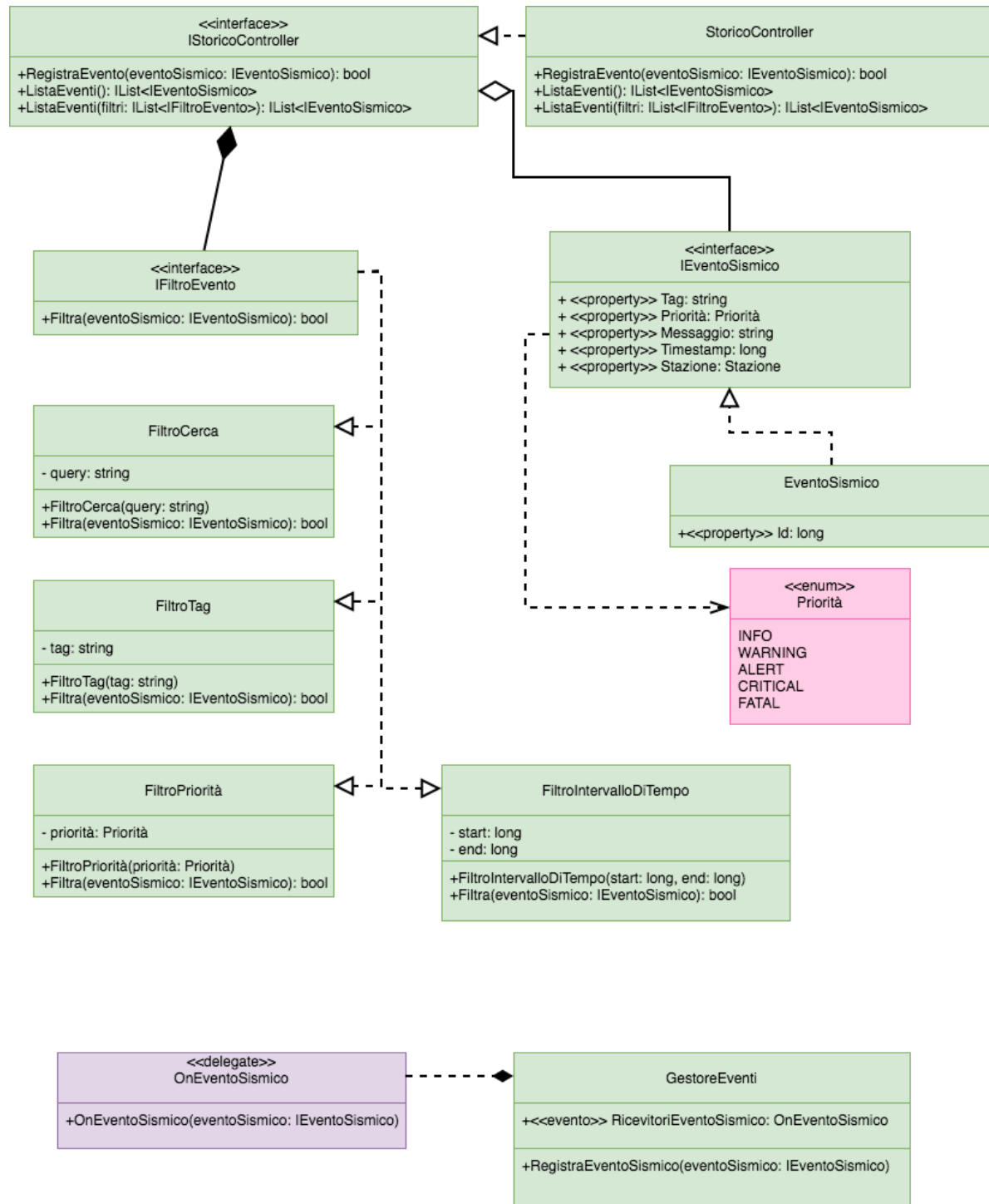
DBController
<ul style="list-style-type: none">- PercorsoDB: string- Connessione: SQLiteConnection
<pre>#DBController(PercorsoDatabase: string) #<<destructor>> DBController() - ApriConnessioneDB() - ChiudiConnessioneDB()</pre>

Questo package si occupa di realizzare i meccanismi di accesso al database *SQLite*.

DBController

Questa classe si occupa di aprire e chiudere una connessione al database *SQLite* specificato. Tutte le classi derivate potranno quindi accedere al database senza dover esplicitamente iniziare o terminare l'interazione.

Package io.sismio.evento



In questo package sono racchiuse tutte le classi inerenti al modello degli eventi sismici, alla loro persistenza e alla gestione delle notifiche relative.

IStoricoController

Interfaccia che rappresenta il concetto di archivio persistente degli eventi sismici.

StoricoController

Estende DBController e implementa IStoricoController. Permette di salvare gli eventi sismici in maniera persistente sul database e ne permette inoltre la successiva lettura.

IEventoSismico

Interfaccia che rappresenta il concetto di evento sismico e fornisce i metodi per leggere e scrivere le sue proprietà. Da notare come in fase di progettazione si è ritenuto necessario trasformare la data in un timestamp numerico, per semplificarne la gestione.

EventoSismico

Implementazione di IEvento, permette di disaccoppiare l'interfaccia di evento sismico alla sua rappresentazione nel database. In particolare, viene aggiunto un attributo ID che rappresenta l'identificatore surrogato della entry relativa all'evento. Nascondendolo dietro ad un'interfaccia, l'utente finale non si accorge dei campi relativi alla persistenza.

Priorità

Enum che rappresenta tutti i tipi possibili di priorità di un certo evento sismico.

IFiltroEvento

Interfaccia che rappresenta il concetto di filtro, utile per ricercare una lista di eventi secondo un certo criterio. Deve essere implementata tramite un pattern **Strategy** definendo una strategia per ogni criterio di filtraggio. Le classi derivate definiscono un metodo `Filtra()` che prende in ingresso un evento sismico e, tramite il valore di ritorno, determina se l'evento rispetta o meno il criterio specificato. Utilizzando questo pattern è inoltre molto facile concatenare più criteri di scelta.

FiltroCerca

Implementa una strategia di ricerca prendendo in ingresso una stringa di query. Restituisce vero se l'evento contiene la query specificata nel messaggio.

FiltroTag

Implementa una strategia di filtraggio basandosi sul campo tag di un evento. Restituisce vero se il tag dell'evento equivale a quello specificato.

FiltroPriorità

Implementa una strategia di filtraggio basandosi sul campo priorità di un evento e restituisce vero per gli eventi con priorità uguale a quella specificata.

FiltroIntervalloDiTempo

Implementa una strategia di filtraggio basandosi sul campo timestamp di un evento. Prende in ingresso un intervallo temporale, verificando se l'evento è all'interno di esso.

GestoreEventi

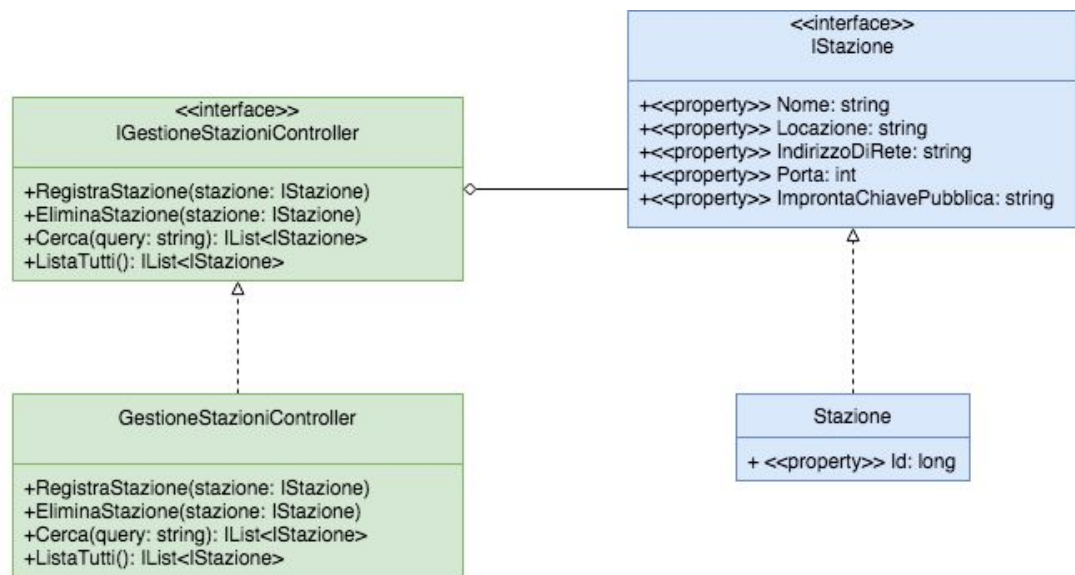
Viene utilizzato da ogni componente che ha necessità di registrare un nuovo evento o di essere notificato. Il metodo per registrare un evento deve essere implementato in modo tale da risolvere eventuali corse critiche relative alla concorrenza multi-thread dell'analisi.

Nel momento in cui un componente registra un evento, il `GestoreEventi` si occupa di inoltrarlo a tutti i ricevitori che si sono registrati. Questo permette di far passare un evento generato dall'Analisi a qualunque parte del sistema che ne ha bisogno, come ad esempio il sistema di notificazione.

`OnEventoSismico`

Delegate che rappresenta l'evento di una scossa sismica. Viene utilizzato dal `GestoreEventi` e permette a tutti i ricevitori di eventi di registrarsi.

Package `io.sismio.stazione`



Questo package si occupa di gestire le stazioni remote, registrandole opportunamente su database.

IStazione

Interfaccia che modella il concetto di Stazione. In particolare un campo mantiene l'impronta digitale della chiave pubblica del certificato necessaria alla connessione sicura.

Stazione

Implementazione di IStazione, permette di disaccoppiare l'interfaccia di una stazione remota alla sua rappresentazione nel database. In particolare, viene aggiunto un attributo ID che rappresenta l'identificatore surrogato della entry relativa alla stazione. Nascondendolo dietro ad un interfaccia, l'utente finale non si accorge dei campi relativi alla persistenza.

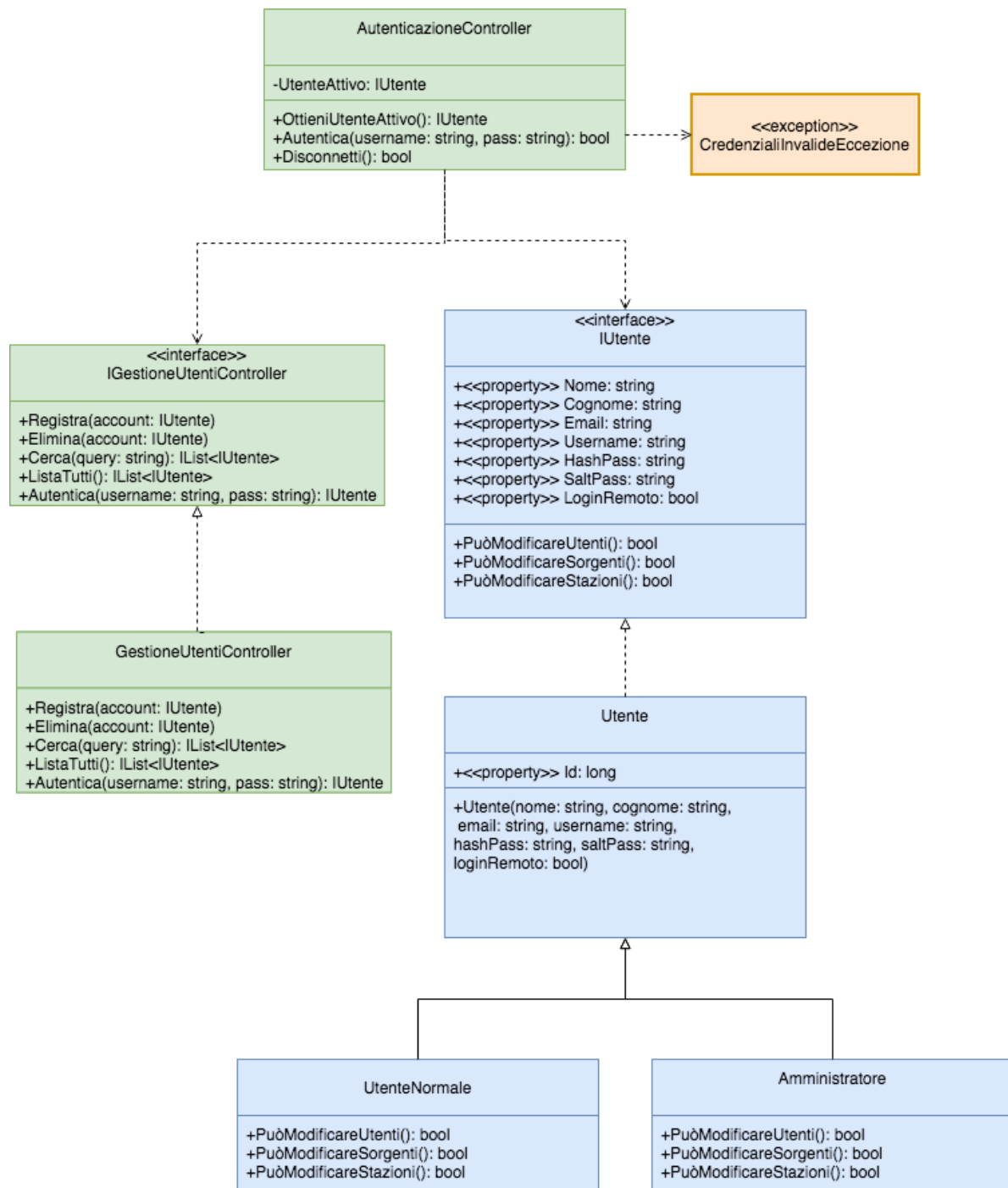
IGestioneStazioniController

Interfaccia che rappresenta il concetto di archivio di stazioni. Astrae la gestione rendendola indipendente dal supporto fisico usato per la persistenza. Mantiene una composizione di IStazione, intesa come collezione riassunta in una IList.

GestioneStazioniController

Implementa IGestioneStazioniController ed estende DBController. Specifica i comportamenti dell'interfaccia nell'ambito di archiviazione tramite database, in particolare attraverso tecnologia *SQLite* per la memorizzazione persistente dei dati. Le funzionalità di connessione al database derivano dall'estensione di DBController, le quali vengono utilizzate per concretizzare la logica di business.

Package io.sismio.utente



Questo package si occupa della gestione degli utenti e delle loro sessioni. Permette inoltre di gestirne la persistenza su database.

Breve considerazione sulla sicurezza

Per motivi di sicurezza, sono stati aggiunti `HashPass` e `SaltPass` all'utente.

`HashPass` rappresenta l'impronta, l'"hash" della password, ottenuta grazie ad una cifratura di questa attraverso algoritmo SHA256.

`SaltPass` rappresenta invece una sequenza di caratteri casuali che viene concatenata alla password vera e propria: questo meccanismo permette di aumentare la complessità di calcolo nel caso di attacchi informatici e aumenta quindi il grado di sicurezza.

`UtenteNormale` rappresenta il privilegio `SEMPLICE`, `Amministratore` rappresenta il privilegio `AVANZATO`.

`IGestioneUtentiController`

Interfaccia che rappresenta il concetto di archivio di utenti. Astrae la gestione rendendola indipendente dal supporto fisico usato per la persistenza. Mantiene una composizione di `IUtente`, intesa come collezione riassunta in una `IList`. Definisce inoltre il comportamento di autenticazione espresso dal metodo `Autentica`.

`GestioneUtentiController`

Implementa `IGestioneUtentiController` ed estende `DBController`. Specifica i comportamenti dell'interfaccia nell'ambito di archiviazione tramite database, in particolare attraverso tecnologia `SQLite` per la memorizzazione persistente dei dati. Le funzionalità di connessione al database derivano dall'estensione di `DBController`, le quali vengono utilizzate per concretizzare la logica di business. Implementa il comportamento `Autentica` attuando la lettura delle credenziali dal sistema di persistenza e confrontandole con quelle ricevute in ingresso.

`AutenticazioneController`

Gestore degli account univoco all'interno del sistema. Permette di autenticare un utente alla stazione e di reperire quello attivo correntemente. Attraverso `Disconnetti()` è possibile terminare la sessione di lavoro dalla stazione corrente.

`IUtente`

Rappresenta l'interfaccia di un generico utente. Oltre a definire i campi principali, definisce 3 metodi che permettono in maniera semplice di determinare il grado di permessi di una particolare sottoclasse. In particolare, le classi derivate dovranno implementare questi tre metodi per restituire vero o falso in base alla possibilità di eseguire quella determinata azione.

`Utente`

Classe astratta che rappresenta il concetto di utente. Raccoglie tutte le proprietà comuni ai diversi tipi di utente. Mantiene astratti i metodi relativi ai privilegi affinché le classi derivate li implementino per definire i livelli di permessi in base alla specializzazione della classe (privilegi utente).

`UtenteNormale`

Classe derivante da `Account`, rappresenta un utente normale e possiede privilegi di tipo `SEMPLICE`.

`Amministratore`

Classe derivante da `Account`, rappresenta un amministratore e possiede privilegi di tipo `AVANZATO`.

`CredenzialiInvalideEccezione`

Eccezione lanciata dal metodo `Autentica()` nel caso in cui le credenziali di accesso non

siano valide.

Package `io.sismio.log`

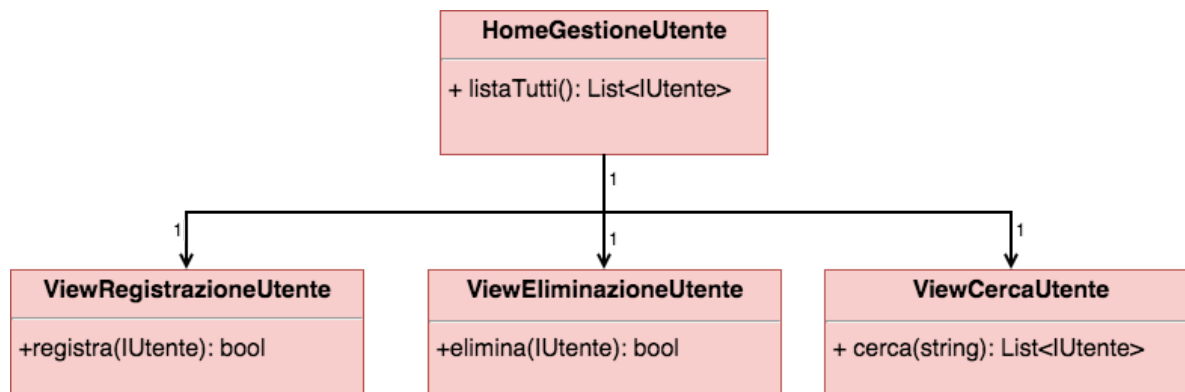
Logger
- FileDiLog: File
+Scrivi(utente: IUtente, messaggio: string): int

Questo package racchiude le classi necessarie a eseguire il log attraverso tutto il sistema.

Logger

Singleton pubblico reperibile da qualsiasi altro componente. Si occupa di scrivere su un file di testo le principali azioni eseguite dagli utenti.

Diagramma di Dettaglio - GestioneUtenti

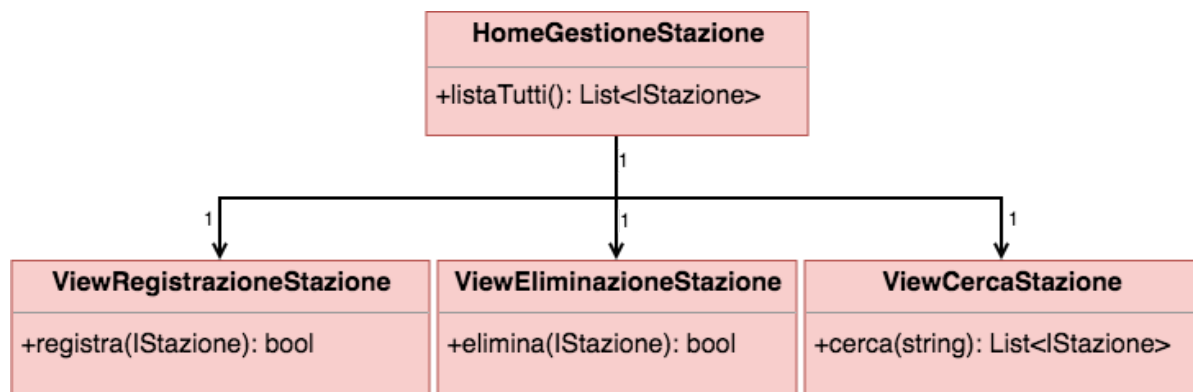


Nel diagramma sopra rappresentato, sono evidenziate tutte le interfacce disponibili riguardanti la gestione degli utenti. Ogni View mette a disposizione una form che permette l'inserimento dei dati utili per lo svolgimento del suo compito. Nella Home è possibile ottenere la lista di tutti gli utenti relativi alla stazione. I campi mostrati relativi agli utenti (email, username, tipo e remote login) sono di sola lettura.

The screenshot shows a web application interface for managing accounts. It includes a sidebar with navigation icons, a header with a search bar and a 'Cerca qui' button, and a main table displaying a list of users. Each user entry includes their email, username, type, and a checkbox for 'Remote login'. A red trash icon is present next to each row, indicating a delete function. A blue circular button with a pencil icon is located at the bottom right of the interface.

Email	Username	Tipo	Remote login	
a@example.it	a-user	user	<input type="checkbox"/>	
b@example.it	b-user	user	<input checked="" type="checkbox"/>	
admin@example.it	c-admin	admin	<input checked="" type="checkbox"/>	
d@example.it	d-user	user	<input type="checkbox"/>	

Diagramma di Dettaglio - GestioneStazioni



Nel diagramma sopra rappresentato, sono evidenziate tutte le interfacce disponibili riguardanti la gestione delle stazioni.

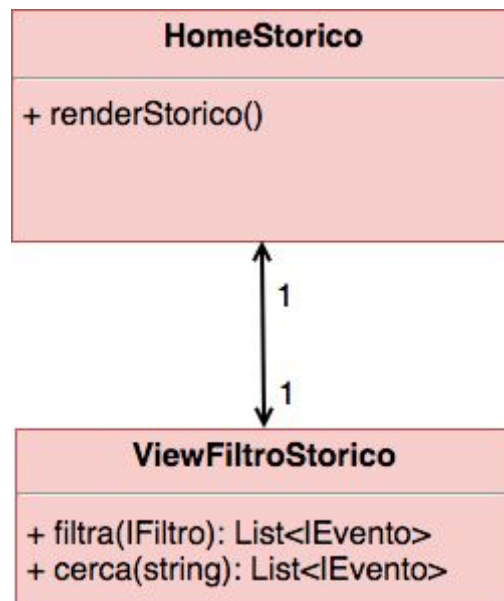
Ogni View mette a disposizione una form che permette l'inserimento dei dati utili per lo svolgimento del suo compito.

Nella Home è possibile ottenere la lista di tutte le stazioni.

The screenshot shows the 'Gestione stazioni remote' web application. It features a dark blue sidebar with navigation icons (home, list, add, settings, users, and a document icon). The main content area has a title 'Gestione stazioni remote' and two buttons: 'Associa nuova' (green) and 'Cerca qui' (blue with a magnifying glass icon). Below the buttons is a table with five columns: 'Nome', 'Stazione d'origine', 'IP : porta', 'Chiave pubblica', and an action column with red delete icons. The table contains four rows of data, with the third row (RS3) highlighted in blue. A blue circular button with a pencil icon is located in the bottom right corner.

Nome	Stazione d'origine	IP : porta	Chiave pubblica	
BS01	Bologna	10.5.23.56:4567	abjduhg3jjAHbd...	
MS01	Modena	10.11.54.2:1235	JDb3isn3jjAHbd...	
RS3	Reggio Emilia	12.9.56.23:6729	Bsdo2sn3jJAHbd...	
CS4	Cesena	10.9.9.5:5423	B7kKo2sn3jJAHbd...	

Diagramma di Dettaglio - Storico



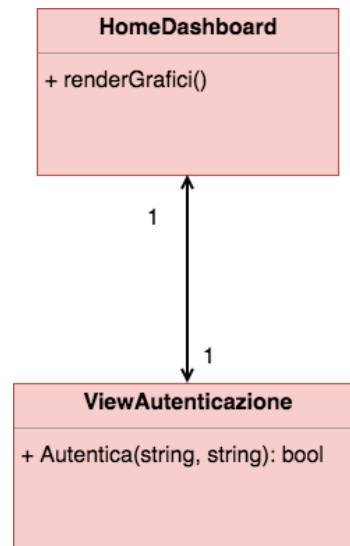
Nel diagramma sopra rappresentato, sono evidenziate tutte le interfacce disponibili riguardanti lo storico.

Ogni View mette a disposizione una form che permette l'inserimento dei dati utili per lo svolgimento del suo compito.

Nella Home è possibile ottenere il render dello storico.

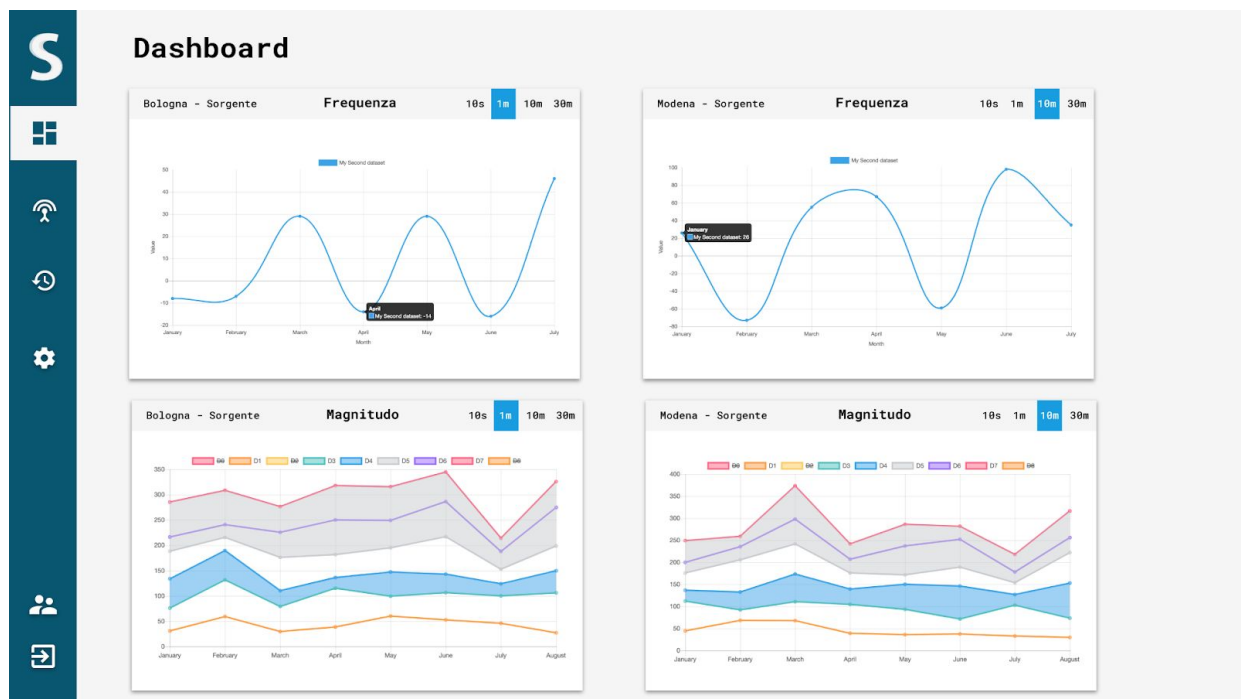
<div>S</div> <div>☐</div> <div>📶</div> <div>🕒</div> <div>⚙️</div> <div>👤</div> <div>📄</div>	Storico eventi		Cerca qui 🔍	Filtro eventi sismici		Priorità	INFO
				Inizio	06-05-2018 14:00	Fine	08-05-2018 20:00
	Data e ora	Tag	Stazione d'origine	Messaggio		Priorità	
	08-05-2018 19:56:18	FREQUENZA	Bologna - S1	Brevi picchi negli ultimi 15 m...		INFO	
	08-05-2018 14:56:18	MAGNITUDO	Modena - S1	Scossa breve magnitudo 2		ALERT	
	07-05-2018 11:56:18	FREQUENZA	Bologna - S1	Frequenti scosse nell'ultimo m...		WARNING	
	06-05-2018 14:56:18	MAGNITUDO	Modena - S1	Scossa magnitudo 5 prolungata ...		CRITICAL	

Diagramma di Dettaglio - Dashboard



Nel diagramma sopra rappresentato, sono evidenziate tutte le interfacce disponibili riguardanti la dashboard.

Nella Home è possibile ottenere il render dei grafici.



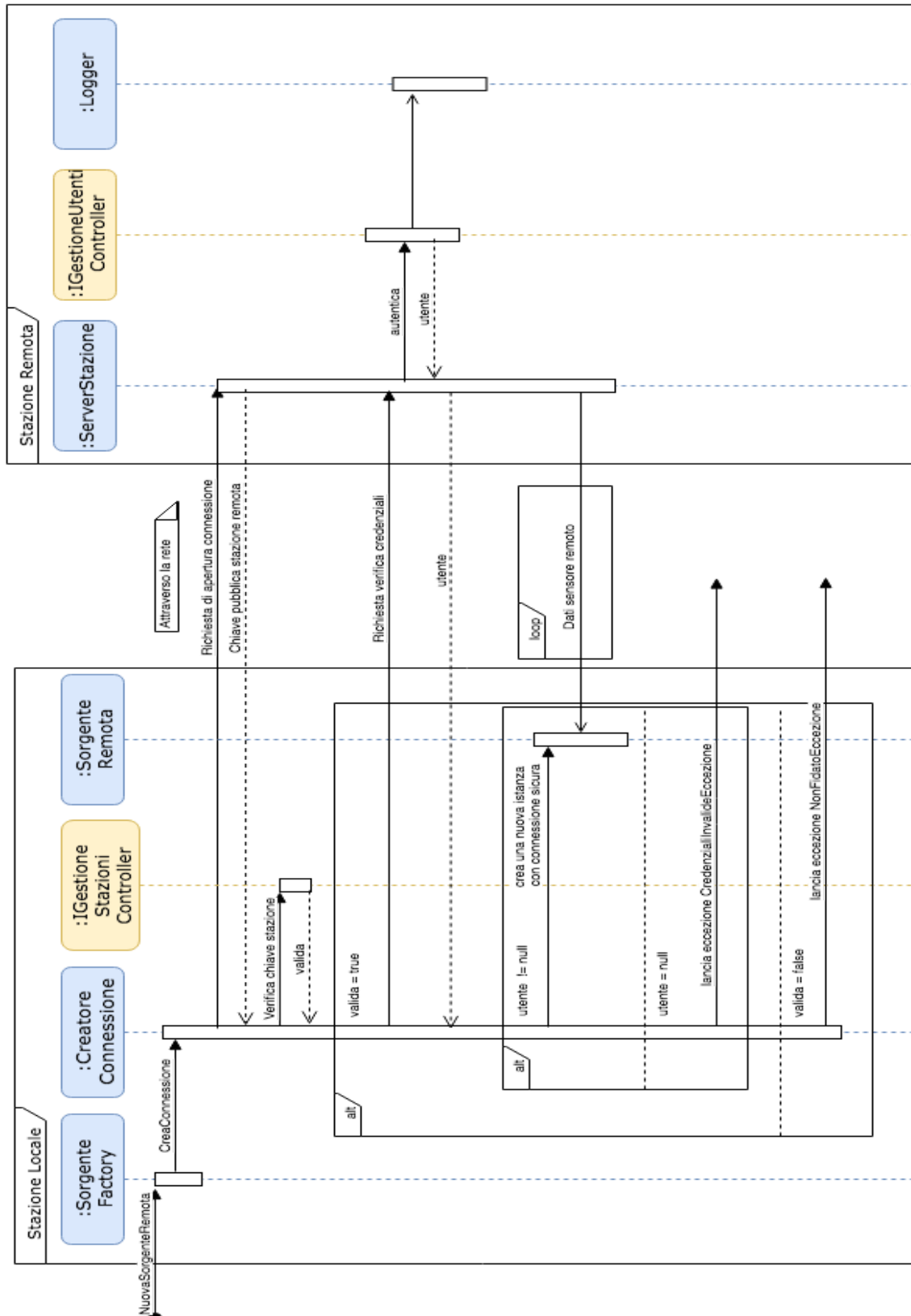
ViewAutenticazione mette a disposizione una form che permette l'inserimento dei dati utili per lo svolgimento del suo compito.

The image shows a login form for a system named 'Sismio'. The form is centered on a dark teal background. It consists of three vertically stacked white rectangular input fields. The first field is labeled 'Username', the second 'Password', and the third is a blue button labeled 'ACCEDI'. The text 'Sismio' is displayed in a large, white, sans-serif font above the input fields.

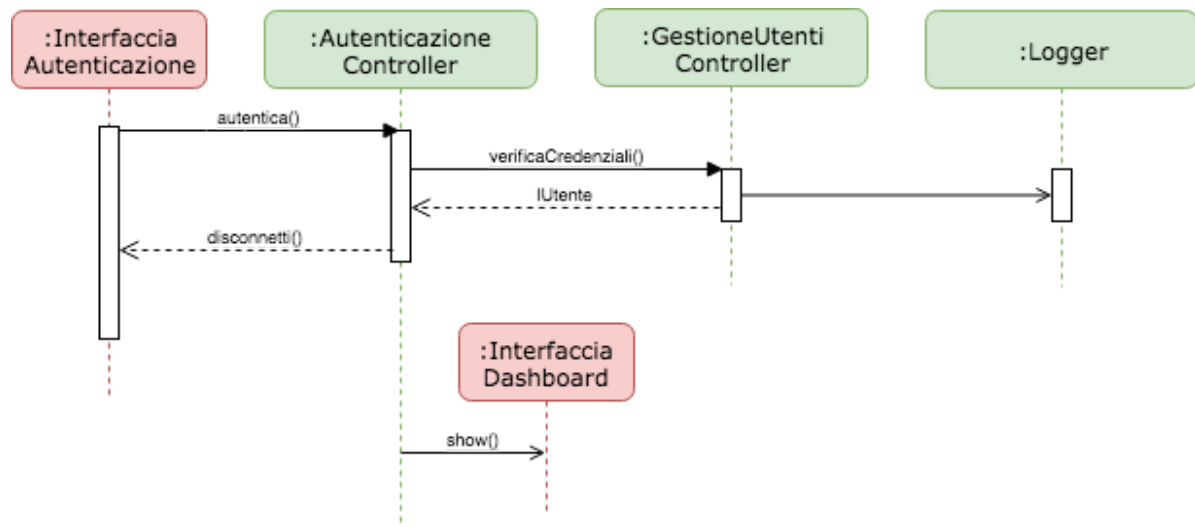
Sismio

Interazione

Creazione della connessione sicura



Autenticazione ad una stazione locale



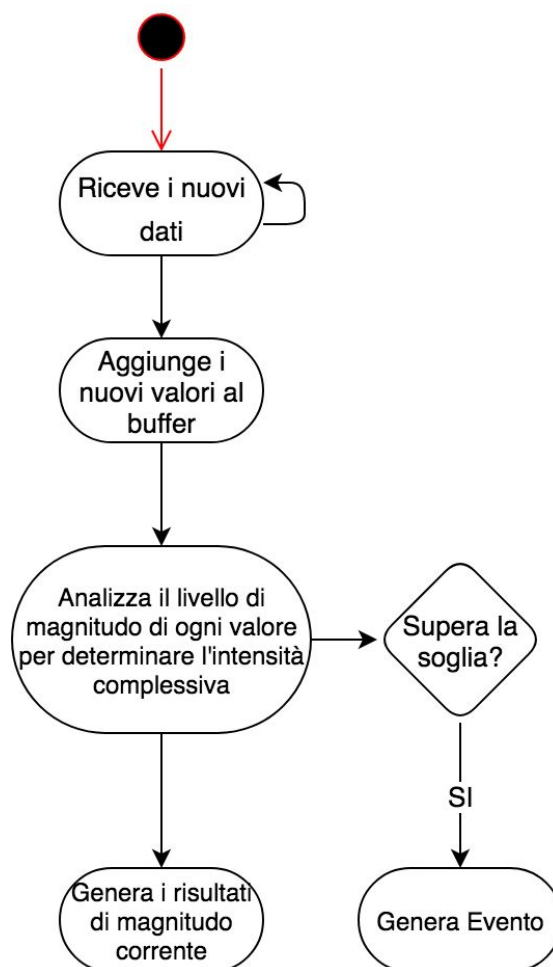
Comportamento

Come nella parte di analisi del problema, si è ritenuto che non esistessero entità con uno stato tale da rendere necessaria la creazione di un diagramma di stato.

Si è tuttavia deciso di inserire i diagrammi delle attività dei due algoritmi utilizzati per analizzare i valori del sensore, ovvero magnitudo e frequenza.

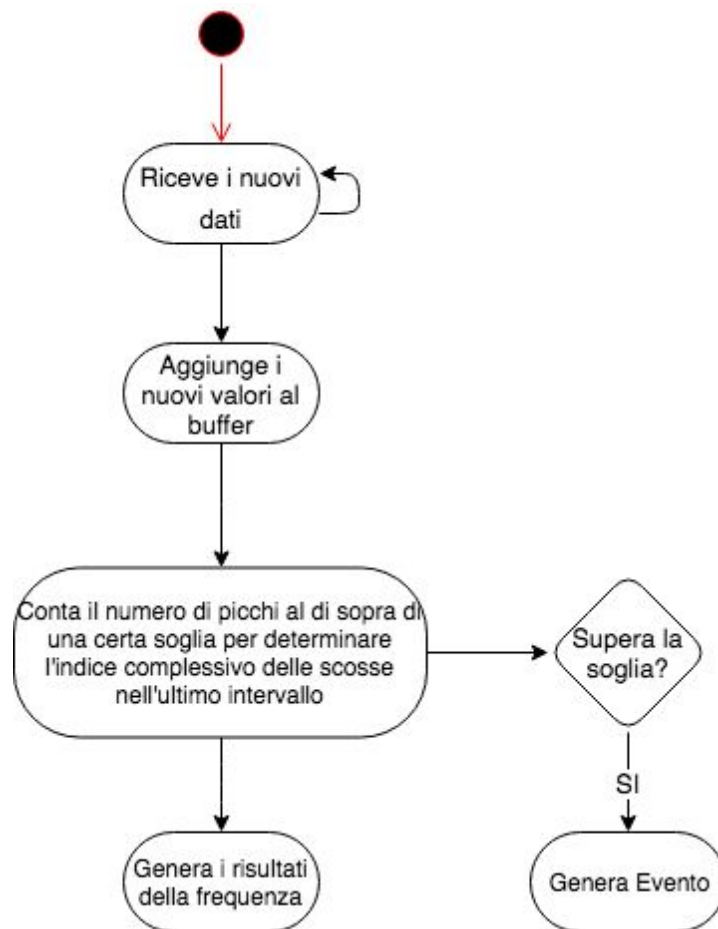
Algoritmo Analisi Magnitudo

L'analisi della magnitudo permette di determinare il valore di magnitudo nell'ultimo intervallo di tempo. Per farlo, analizza ogni valore del buffer per determinare la magnitudo complessiva e, se supera una certa soglia, genera un evento sismico.

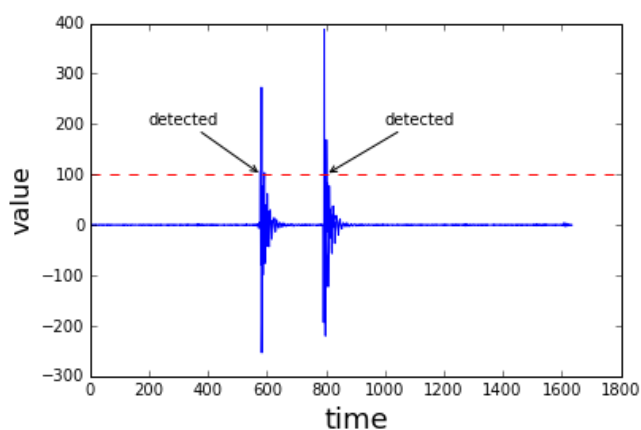


Algoritmo Analisi Frequenza

L'analisi della frequenza permette di determinare quante scosse sismiche si sono verificate nell'ultimo istante di tempo. Per farlo, analizza il buffer dei valori e conta il numero di campioni che superano una certa soglia. Questo conteggio rappresenta l'indice di frequenza delle scosse sismiche.



Il grafico sottostante fa vedere un tipico caso di scossa sismica che risulterebbe in un indice di frequenza sismica pari a 2.



Persistenza

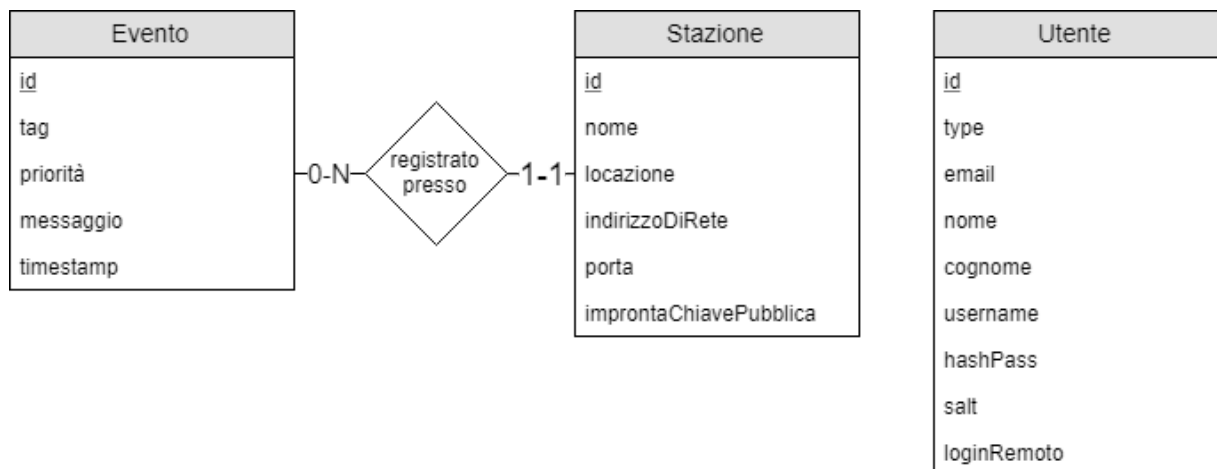
Il seguente diagramma ER rappresenta le entità e le relazioni appartenenti alla persistenza.

Entità

- Evento: ogni evento è identificato da un id (chiave primaria) e presenta gli attributi evidenziati
 - Stazione: ogni stazione è identificata da un id (chiave primaria) e presenta gli attributi evidenziati.
 - Utente: ogni utente è identificato da un id (chiave primaria) e presenta gli attributi evidenziati. Inoltre, **l'email e l'username devono risultare univoci all'interno del sistema.**
- ☐ Le chiavi primarie sono di tipo intero, autoincrement
 - ☐ Utente: type è necessario in quanto nella fase di programmazione, nel momento in cui viene creata l'istanza relativa, bisogna identificare e salvare il privilegio dell'account (SEMPLICE o AVANZATO)

Relazioni

- Event (0..N) - (1-1) Stazione: ogni evento è registrato presso una sola stazione; ogni stazione può avere registrati 0 o più eventi



Sicurezza

Le problematiche di sicurezza relative al database sono state esposte nella parte di "Considerazioni sulla sicurezza relative alle tecnologie utilizzate".

Formato del file di log

Il file di log deve contenere tutte le azioni svolte da un utente (è possibile effettuare l'operazione di scrittura sul file di log senza specificarlo) all'interno del sistema, con la relativa data e ora. Il formato del file è il seguente:

```
<timestamp> <username> <messaggio>  
<timestamp> <username> <messaggio>  
<timestamp> <username> <messaggio>
```

```
<timestamp> <messaggio>  
<timestamp> <messaggio>  
<timestamp> <messaggio>
```

Protezione del file di log

Vista la grande importanza del file di log, è necessario proteggerlo da eventuali manomissioni da parte di attaccanti esterni. A questo scopo vengono utilizzate le funzionalità del sistema operativo che permettono di controllarne gli accessi: un esempio può essere quello di impostare i permessi di lettura solo all'utente Amministratore, garantendo così la sicurezza richiesta. Nessun altro utente della stazione potrà dunque leggere il file indicato; è necessario dunque che esista un utente di tipo Amministratore* nel sistema operativo di ogni stazione.

Vengono inoltre eseguiti backup periodici in un server remoto in maniera da garantirne la disponibilità anche dopo molto tempo.

* Per Amministratore si intende l'utente che possiede i privilegi AVANZATI, ovvero colui che si occupa della gestione dell'intero sistema.

Progettazione del collaudo

Partendo dai risultati ottenuti dal piano di collaudo dell'architettura logica sono stati sviluppati ulteriori test per verificare il corretto funzionamento delle nuove parti del sistema. Per brevità vengono riportati solo i più caratteristici:

```
[TestClass]
public class TestGestoreEventi
{
    public static IEventoSismico creaEventoMock()
    {
        return new EventoSismico("Magnitude", Priorita.FATAL, "Sorry
brother", 123456, null);
    }

    [TestMethod]
    public void TestGestoreEventi()
    {
        // Crea un mock di Evento
        IEventoSismico e = creaEventoMock();

        // Crea un mock di StoricoController
        Mock<IStoricoController> storicoController = new
Mock<IStoricoController>();

        GestoreEventi gestoreEventi = new
GestoreEventi(storicoController.Object);

        // Registra due ricevitori di evento e assicurati che vengano
chiamati
        bool primoRicevuto = false;
        bool secondoRicevuto = false;

        gestoreEventi.RicevitoriEventoSismico += (evnt) =>
        {
            // Assicurati che l'evento ricevuto sia quello giusto
            Assert.AreEqual(e, evnt);
            primoRicevuto = true;
        };

        gestoreEventi.RicevitoriEventoSismico += (evnt) =>
        {
            // Assicurati che l'evento ricevuto sia quello giusto
            Assert.AreEqual(e, evnt);
            secondoRicevuto = true;
        };
    }
}
```



```

    };

    // Registro l'evento con il gestore di eventi
    gestoreEventi.RegistraEvento(e);

    // Assicurati che siano stati ricevuti
    Assert.IsTrue(primoRicevuto);
    Assert.IsTrue(secondoRicevuto);

    // Assicurati che lo StoricoController sia stato chiamato
    storicoController.Verify(x => x.RegistraEvento(e), Times.Once());
}
}

```

```

[TestClass]
public class TestFiltroEvento
{
    [TestMethod]
    public void TestFiltroPriorita()
    {
        IFiltroEvento filtro = new FiltroPriorita(Priorita.INFO);
        IEventoSismico e = TestGestoreEventi.creaEventoMock();

        Assert.IsFalse(filtro.Filtra(e));

        IFiltroEvento filtro2 = new FiltroPriorita(Priorita.FATAL);
        Assert.IsTrue(filtro2.Filtra(e));
    }

    [TestMethod]
    public void TestFiltroCerca()
    {
        IEventoSismico e = TestGestoreEventi.creaEventoMock();

        IFiltroEvento filtro = new FiltroCerca("yeah");
        Assert.IsFalse(filtro.Filtra(e));

        IFiltroEvento filtro2 = new FiltroCerca("BRO"); // Test case
insensitive
        Assert.IsTrue(filtro2.Filtra(e));

        IFiltroEvento filtro3 = new FiltroCerca("gnitude"); // Test case
insensitive
        Assert.IsTrue(filtro3.Filtra(e));
    }
}

```

```

    }

    [TestMethod]
    public void TestFiltroTag()
    {
        IEventoSismico e = TestGestoreEventi.creaEventoMock();

        IFiltroEvento filtro = new FiltroTag("Magnitude");
        Assert.IsTrue(filtro.Filtra(e));

        IFiltroEvento filtro2 = new FiltroTag("Frequency");
        Assert.IsFalse(filtro2.Filtra(e));
    }

    [TestMethod]
    public void TestFiltroIntervalloDiTempo()
    {
        IEventoSismico e = TestGestoreEventi.creaEventoMock();

        IFiltroEvento filtro = new FiltroIntervalloDiTempo(123000,
123999);
        Assert.IsTrue(filtro.Filtra(e));

        IFiltroEvento filtro2 = new FiltroIntervalloDiTempo(123999,
999999);
        Assert.IsFalse(filtro2.Filtra(e));

        IFiltroEvento filtro3 = new FiltroIntervalloDiTempo(123000,
123123);
        Assert.IsFalse(filtro3.Filtra(e));
    }
}




```

Piano del deployment

L'applicazione è volutamente autocontenuta ed ogni nodo si comporta sia da cliente che da servitore. Il deployment deve quindi essere semplice per venire incontro alle esigenze del cliente. L'unico requisito del sistema è che la macchina sia connessa ad internet e che un eventuale firewall permetta le connessioni dall'esterno sulla porta dell'applicazione.

Essendo ogni nodo a se stante, il proprietario di una macchina è anche l'amministratore e ha quindi la responsabilità di gestire utenti, sensori ed eventuali stazioni remote. Per rendere il lavoro di manutenzione più facile, nella versione futura dell'applicazione (2.0) sono previste diverse schermate che permettono all'amministratore di verificare lo stato corrente del sistema, gli utenti correntemente connessi e per gestire eventuali aggiornamenti. Le figure sottostanti illustrano come le varie interfacce appariranno in futuro.

Utenti remoti connessi

Username	Durata sessione	
pippo	da 3 ore	
pluto	da 5 minuti	
paperino	da 4 ore e 30 minuti	

La maschera degli utenti remoti connessi permette all'amministratore di vedere quali utenti sono correntemente connessi al sistema e da quanto tempo. Permette inoltre di invalidare manualmente la sessione di un utente remoto in caso di irregolarità.

Stato del Sistema

Internet	 attivo
Sensore	 non attivo
Numero utenti remoti	8
Sistema di allarme	 attivo

La maschera rappresenta lo stato attuale del sistema, riferito alla stazione d'interesse. E' possibile ottenere informazioni riguardo Internet, l'effettiva funzionalità del sensore, il numero di utenti remoti collegati alla stazione e l'attività del sistema d'allarme.

Aggiornamenti

Versione	Stato
patch sicurezza (2.1)	<i>installato</i>
patch prestazioni (2.2)	installato
patch bug fix (2.3)	installa

Questa maschera permette di visualizzare gli ultimi aggiornamenti effettuati e quelli disponibili. L'amministratore può dunque cliccare sul pulsante installa per avviare la procedura di aggiornamento. Dopo l'aggiornamento è necessario un riavvio del sistema appena possibile.

Implementazione

Nella fase di implementazione sono stati effettuati una serie di piccoli cambiamenti dovuti all'utilizzo del framework .NET. Un resoconto delle scelte implementative è descritto nel documento "Rapporto sull'Implementazione". In seguito vengono solo elencate le principali differenze tra progettazione e codice definitivo:

- Cambiato il tipo di risultato notificato da `OnRisultatoAnalisi`, che adesso non è più un array ma bensì un valore singolo in quanto la libreria di grafici scelta supporta autonomamente la bufferizzazione. La scelta inoltre migliora notevolmente le performance.
- Introdotti in `EventoSismico`, `Utente` e `Stazione` i metodi "`ConvertiRigaIn...`" che accettano come parametro il cursore di una query sql, effettuano il parsing ed il popolamento dei campi.
- Aggiunta la classe `MockSensore`, che permette di simulare un sensore locale durante la fase di testing.
- Aggiunta all'interfaccia `ISorgente` il metodo `RimuoviAnalisi`, utile in fase di rilascio delle risorse.
- Nella classe astratta `Sorgente`, rinominato il metodo `OnDatiDisponibili` in `NotificaDatiDisponibili` per evitare ambiguità.
- Aggiunti i riferimenti a `GestoreEventi` e `IStazione` nelle `Sorgenti` e nelle `Analisi`, necessari in fase di creazione e notificazione di un `EventoSismico`.
- Aggiunto il metodo `ValidaCertificato()` nel `GestoreStazioniController`, necessario per verificare che il certificato passato come parametro appartenga ad una stazione remota registrata.
- Passata un'istanza di `GestioneStazioniController` al `CreatoreConnessioni`, che la utilizza per validare la stazione remota.
- Sostituito il distruttore della classe `DBController` con `Dispose`, per realizzare una finalizzazione deterministica.
- Aggiunta l'eccezione `RispostaInvalidaEccezione` per segnalare un fallimento nell'handshake iniziale verso una stazione remota.
- Passati i riferimenti di `Sensore` e `GestioneUtentiController` a `ServerStazione`, necessari rispettivamente alla trasmissione dei dati e all'autenticazione remota.
- Per realizzare il meccanismo della concorrenza sono state aggiunte una serie di code bloccanti che permettono ai vari thread di comunicare. Su questa parte vengono date ulteriori informazioni nel documento "Rapporto sull'implementazione".
- Passato in ingresso a `TrasmissioneDatiWorker` un `GestioneUtentiController`, che verrà poi utilizzato per effettuare l'autenticazione remota.
- In `IGestioneStazioniController` e nella sua implementazione, modificato leggermente i nomi dei metodi (funzionalità rimangono le stesse), `Registra` ed `Elimina` restituiscono un bool invece che void e viene passata una stazione invece che la sua interfaccia (necessità id), package stazione
- Aggiunto override del metodo `Scrivi` nel package log: ora non è più necessario specificare l'utente

Deployment

Deployment Specification

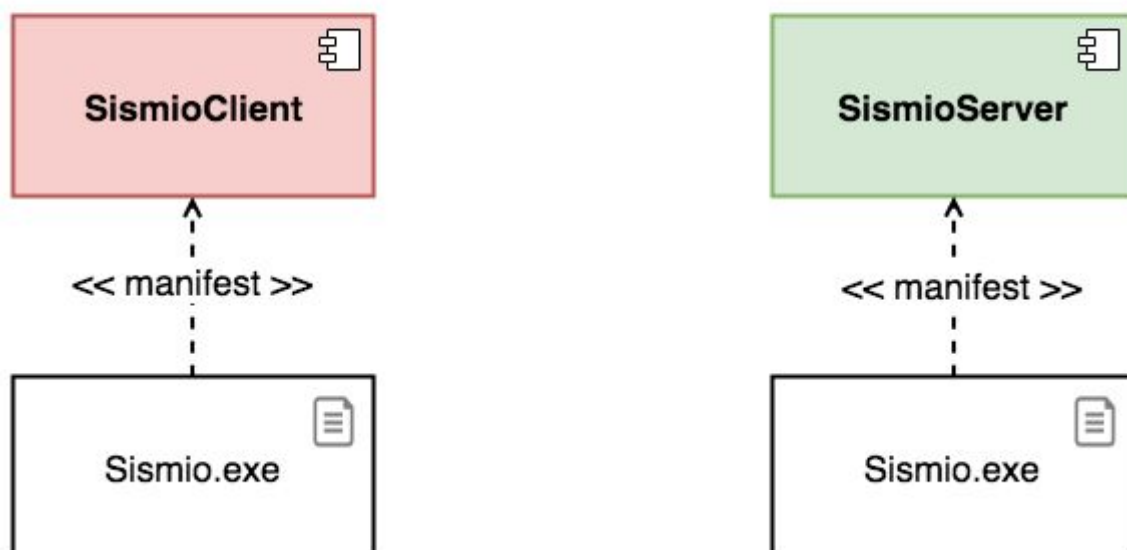
Per configurare correttamente un nodo, è necessario modificare il file *SensoreConf.xml* in modo da poter leggere correttamente da un sensore:



Inoltre è necessario configurare correttamente la chiave privata per garantire una connessione sicura. Per farlo si utilizzano gli strumenti forniti dal sistema operativo *Windows* che permettono di creare un file *PFX* contenente la chiave privata autofirmata.

Artefatti

Il sistema Sismio racchiude i componenti cliente e servere generando un unico artefatto



Deployment Type-Level

L'applicazione è auto-contenuta, per questo motivo il nome dell'eseguibile è uguale, comportandosi da servitore e/o da cliente.

